Theoretical Computer Science 77 (1990) 131-159
North-Holland

EQUATIONAL TYPE LOGIC

V. MANCA and A. SALIBRA
Dip. Informatica, University of Pisa, Corso Italia 40, I-56100 Pisa, Italy

G. SCOLLO
Dept. Informatica, University of Twente, P.O. Box 217, NL-7500AE Enschede, Netherlands

Abstract. Equational type logic is an extension of (conditional) equational logic, that enables one to deal in a single, unified framework with diverse phenomena such as partiality, type polymorphism and dependent types. In this logic, terms may denote types as well as elements, and atomic formulae are either equations or type assignments. Models of this logic are type algebras, viz. universal algebras equipped with a binary relation—to support type assignment. Equational type logic has a sound and complete calculus, and initial models exist. The use of equational type logic is illustrated by means of simple examples, where all of the aforementioned phenomena occur. Formal notions of reduction and extension are introduced, and their relationship to free constructions is investigated. Computational aspects of equational type logic are investigated in the framework of conditional term rewriting systems, generalizing known results on confluence of these systems. Finally, some closely related work is reviewed and future research directions are outlined in the conclusions.

1. Introduction

Many-sorted (conditional) equational logic is the most established basis to the algebraic approach to abstract data type (ADT) specification [15, 9]. Heterogeneous algebras [17, 4] are the standard models of this logic, extending universal algebra structures in a straightforward way. In algebraic specification, however, several phenomena indicate that this logic encounters limitations in practice. We mention a few, most interesting of these phenomena (which are discussed in Section 2): partiality, exception handling, extension, type polymorphism, dependent types.

Several formal frameworks have been designed to solve the problems that are raised by *each* of these phenomena. In particular, many of these frameworks are based on extensions of equational logic in various forms. Most of these approaches address the phenomena of their interest at a rather high level of generality. Yet, a unifying approach, where all of these phenomena can be dealt with, does not seem to have emerged.

The following problem is addressed in this paper: to find and investigate a *parsimonious* logic of types where *all* of the aforementioned phenomena can be dealt with in an algebraic setting. In Section 2 we further motivate our investigation by offering, for each of those phenomena, a short discussion and a simple example. The equational type logic (ET logic) presented in this paper (Sections 3 and 4) is

a generalization of many-sorted equational logic that extends "reasoning with equations" towards "reasoning with equations and type assignments". ET logic has a sound and complete calculus, and initial models exist. In Section 5 we demonstrate the use of ET logic in the simple examples mentioned in Section 2.

Further results are then worked out; they can be summarized as follows. Formal notions of reduction and extension are introduced, and their relationship to free constructions is investigated; the standard adjunction is extended to a complete sublattice of the ET substructure lattice (Section 6). ET rewriting systems are introduced, as a proper extension of conditional term rewriting systems; we show that confluence of ET rewriting systems holds under the same syntactical conditions that were established by [2] for conditional term rewriting systems (Section 7). Here we summarize the main intuitions behind ET logic:

(1) Elements and sorts (or types, which from now on we use synonymously) are *merged* in a single carrier equipped with a binary *typing* relation, which assigns types to elements (hence types are elements themselves). This immediately introduces partiality because, in general, an operation is defined only on elements of suitable types. Moreover, one obtains a great amount of flexibility and generality; several types may be assigned to an element, operations may take type arguments or yield types, etc.

(2) Usual ADT presentations consist of two parts: signature and logical axioms. ET presentations *merge* the type constraints and the equality ones. In fact, ET formulae are conditional formulae where equations and type assignments may occur indifferently in the premise and in the conclusion.

These intuitions were first exploited in [18]: an extension of many-sorted equational logic, in the sense mentioned above, was introduced. The semantics was set in a partial-algebraic framework. The pragmatics of that logic were further investigated in [19]. Solicited by an anonymous referee, and inspired by Mosses' *unified algebras* [22] (see Section 8), we found that an adequate representation of partiality is also available in the semantically simpler framework of *total* one-sorted algebras (thus, standard universal algebras), still equipped with the typing relation. In these algebras, which we call *type algebras* in view of the relevance of the typing relation, the partiality of an operation can be expressed in several ways, for example by letting the undefined applications of the operation yield *underdefined elements* (such is considered any element of the carrier if neither any type is assigned to it nor is itself a type of some element of the carrier).

This choice of semantical simplicity, together with the fact that no restriction is put on the typing relation, resulted in the following, somewhat surprising outcome; formally, ET logic can be viewed as a Horn logic with equality and one (binary) predicate, viz. type assignment. This fact renders the parsimony of this logic very well. However, the obvious conclusion that ET logic *is* a special case of Horn logic can easily prove misleading. Types and predicates are very different logical devices, although either has sufficient expressive power to "encode" the other. The differences, which arise from the intuition that underlies these two formal devices, are

reflected in their "natural" use; for instance, Horn predicates do not have predicate arguments, whilst type arguments in type terms are just natural in ET logic, as mentioned above.

The reader may wonder: Is *any* binary relation acceptable as a typing relation? Our answer is: Absence of restrictions on the typing relation and freedom of *term construction* as a facility to express types are the essential tools by which generality of description is achieved in ET logic. That is, for the general case of the statement "A is of type B", the establishment of different constraints on what A and B can stand for seems arbitrary and unnecessarily restrictive. Differences may well arise in specific contexts, where given purposes and scope of application motivate usefulness or even necessity of restrictions. It seems interesting to investigate the expressibility of such restrictions in the language of ET logic, rather than in the metalanguage.

A final, introductory point of clarity is now in place, relating to the intuitive difference between our "weak" *logic* of types, and the "stronger", more elaborated type *theories* that have foundational purposes. An important case of these is intuitionistic type theory [21], because of its aim of giving foundations to constructive mathematics. The main difference is that the concern with types in ET logic is of an algebraic nature, which is not the case in intuitionistic type theory. This is not meant to say that ET logic is of no interest to foundational studies. On the contrary, the possible relevance of this logic to the study and comparative analysis of foundational theories, as carried out, e.g. in [16] with first-order predicate logic, is an interesting subject of investigation. However, this subject is outside the scope of this paper.

2. On some phenomena in algebraic specification

A glimpse at classes of phenomena that arise in algebraic specification is given below. Not all classes are mutually disjoint. For each class, we give a short discussion, introducing one or two typical problems, of which we state a very simple example. ET formulations of these two examples are deferred to Section 5. Since the literature on these topics is vast, the given references should be viewed just as examples of relevant literature.

2.1. Partiality

Phenomena of this class naturally arise in algebra (e.g. partially defined inverse operations), as well as in programming (possibly non-terminating computations, denotational semantics of programming languages). Extensions of equational logic that cater for partiality have been studied in various ways: either by taking a partial algebra semantics [1, 24], or by representing partiality in a total algebra setting [23]. Model-theoretic approaches are proposed in [6, 7].

Problems. A general question is what to do about "existence" of undefined applications; in first approximation, one would like to dispose of them without doing *anything*. We consider the problem of supporting this *exception by default* principle.

Example. Natural numbers with zero, successor, predecessor: specification by default of undefinedness of predecessor on zero.

2.2. Exception handling

These phenomena can be viewed as forming a subclass of partiality, where explicit tools for designating and detecting errors are to be available: here the previous question assumes that *something* is to be done about exceptions. A good principle is that of introducing *exception classification* as a prerequisite of exception handling (see [23] for further analysis). Error algebras [11] and exception algebras [3] are relevant algebraic frameworks.

Problems. In modular specification, the problem arises of introducing error classification as an extension of the "exception by default" module. Here, "extension" means that underdefined elements may become defined; see below for a more general notion of extension. A related problem is: keeping exception handling local to the module the exceptions stem from [23].

Example. Exception classification of the application of predecessor on zero, relative to the type of the natural numbers.

2.3. Extension

By this we mean the assignment of new types to elements, preserving their construction and existing types (if any). Type extension occurs in several applications: e.g. it is needed in databases to support information-retrieval with multiple classification. The notion of subsorting, or subtyping, is useful to deal with this phenomenon. Order-sorted algebras were introduced by [12], and later generalized to models of order-sorted (Horn) logic by [14]. The HAS-hierarchy [25] shows a close relationship to order-sorted logics.

Problems. A typical problem is the compatibility between extensions over the same type. Compatibility between different forms of exception-handling is a special case: for example, we consider a notion of "error recovery", which might be termed "recovery by extension", that arises when exceptions *cease* to be viewed as such. The problem consists in *putting together* (in the sense of [8]) different views in a consistent way.

Example. Extension of the natural numbers to the integers, compatibly with error classification.

2.4. Type polymorphism

This phenomenon arises whenever "types may vary": with many-sorted algebras, it is formulated by parameterized parameter passing [9] in first-order fashion, whilst it has higher-order form in functional programming.

Problems. Integration of higher-order polymorphism and algebraic specification. This problem is addressed in [5] by taking many-sorted signatures as bases for higher-order, polymorphic term construction. Our approach conversely aims at expressing higher-order polymorphism *within* an algebraic setting. A related problem is that of expressing type polymorphism without variable-binding operators (such as λ-abstraction), which complicate equational deduction.

Example. Algebraic specification of the identity function as a higher-order, polymorphic function.

2.5. Dependent types

So are called those types that depend on values, e.g. bounded stacks, arrays, etc. The approaches proposed in [22] and [23] amalgamate treatment of this phenomenon with the previous ones (see Section 8 for further comparison).

Problems. Algebraic specification of dependent types. By the very nature of this phenomenon, typechecking requires (some form of) evaluation, thus crossing the border between "compile-time" and "run-time".

Example. Specification of the intervals $[0, n]$ on the natural numbers.

3. Type algebras

We introduce the basic notions about type algebras, which are one-sorted total algebras endowed with typing.

Definition 3.1. Let Ω be a one-sorted algebraic signature, i.e. a set of operators each with a number specifying its arity. An *Ω-signed type algebra* A is a pair $\langle \mathsf{A}, :_{\mathsf{A}} \rangle$, with A a one-sorted (total) Ω-algebra and $:_{\mathsf{A}}$ (the *typing*) a binary relation on the carrier $|\mathsf{A}|$ of A.

Example 3.2. The following structure is a type algebra that represents the standard stack ADT. Let L denote a set of data items, L^* the set of finite sequences on L, λ

the empty sequence and $\alpha\beta$ the concatenation of α and β. A single underdefined element "$\perp$" is in the carrier, and strict partiality of the operations is represented (with "$\perp$" representing undefinedness).

$$A =_{\text{def}} \langle \mathsf{A}, :_A \rangle,$$

where

$$\mathsf{A} =_{\text{def}} \langle |\mathsf{A}|, \Omega^{\mathsf{A}} \rangle,$$

where

$$|\mathsf{A}| =_{\text{def}} L \cup L^* \cup \{\text{item}, \text{stack}\} \cup \{\perp\}$$

and Ω^{A} consists of the following operations:

$$\text{empty}^{\mathsf{A}} =_{\text{def}} \lambda,$$

$$\text{item}^{\mathsf{A}} =_{\text{def}} \text{item},$$

$$\text{stack}^{\mathsf{A}} =_{\text{def}} \text{stack},$$

$$\text{push}^{\mathsf{A}}(\alpha, a) =_{\text{def}} \alpha a \text{ if } \alpha \in L^* \text{ and } a \in L, \perp \text{ otherwise},$$

$$\text{pop}^{\mathsf{A}}(\alpha) =_{\text{def}} \beta \text{ if } \alpha = \beta a \text{ for some } a \in L \text{ and } \beta \in L^*, \perp \text{ otherwise},$$

$$\text{top}^{\mathsf{A}}(\alpha) =_{\text{def}} a \text{ if } \alpha = \beta a \text{ for some } a \in L \text{ and } \beta \in L^*, \perp \text{ otherwise},$$

$$:_A =_{\text{def}} \{\langle a, \text{item}\rangle \mid a \in L\} \cup \{\langle \alpha, \text{stack}\rangle \mid \alpha \in L^*\}.$$

Notation 3.3. We use the denomination "Ω:-algebra" as synonym of "Ω-signed type algebra". Similarly, the prefix "Ω:-" is abbreviated form for "Ω-signed type" in other type-algebraic notions. $|A|$ is alternative notation for $|\mathsf{A}|$.

The usual notions from universal algebra are extended to type algebras, with the additional requirements that are induced by the extra structure introduced by the typing.

Definition 3.4. An *Ω:-morphism* from an Ω:-algebra A into an Ω:-algebra B is an Ω-morphism $\phi : \mathsf{A} \to \mathsf{B}$ that respects the typing, i.e. such that if $a_1 :_A a_2$ then $\phi(a_1) :_B \phi(a_2)$.

Definition 3.5. An *Ω:-congruence* on an Ω:-algebra A is a pair $\theta = \langle \equiv_\theta, :_\theta \rangle$ of binary relations on $|A|$ such that:

(i) $\equiv_\theta$ is an Ω-congruence on A;
(ii) if $a \equiv_\theta b$ and $a :_\theta c$, then $b :_\theta c$;
(iii) if $a \equiv_\theta b$ and $c :_\theta a$, then $c :_\theta b$;
(iv) $:_A \subseteq :_\theta$.

Definition 3.6. If $\theta = \langle \equiv_\theta, :_\theta \rangle$ is an Ω:-congruence on A, then we let $[a]_\theta$ denote the congruence class $[a]_{\equiv_\theta}$ and define the *Ω:-quotient* A/θ to be the Ω:-algebra $\langle A/\equiv_\theta, :_{A/\theta} \rangle$ where the typing relation is defined by $[a]_\theta :_{A/\theta} [b]_\theta$ iff $a :_\theta b$.

From Definition 3.5 it follows immediately that the character of $:_{A/\theta}$ as given above is well-defined, viz. it does not depend on the pair of representatives. Subalgebras have the expected definition, with the typing restricted to the carrier of the subalgebra.

Definition 3.7. If $\boldsymbol{A}, \boldsymbol{B}$ are Ω:-algebras, $\boldsymbol{B}$ is an Ω:-*subalgebra* of $\boldsymbol{A}(\boldsymbol{B} \subseteq \boldsymbol{A})$, if B is an Ω-subalgebra of A and $:_{\boldsymbol{B}} = :_{\boldsymbol{A}} \cap |\boldsymbol{B}|^2$.

If Ω is a one-sorted signature and V a set of variables then, as usual, $\mathsf{T}_\Omega(V)$ denotes the one-sorted Ω-algebra of terms on V. The extension of this notion to type algebras is obtained by endowing $\mathsf{T}_\Omega(V)$ with the empty typing.

Definition 3.8. $\boldsymbol{T}_\Omega(V) =_{\text{def}} \langle \mathsf{T}_\Omega(V), \emptyset \rangle$ is the *type algebra of terms* on signature Ω and variables V.

The type algebra of ground terms is the Ω:-subalgebra of $\boldsymbol{T}_\Omega(V)$ that is generated by the empty set of variables.

Definition 3.9. $\boldsymbol{T}_\Omega =_{\text{def}} \boldsymbol{T}_\Omega(\emptyset)$ is the *word* Ω:-*algebra.*

Proposition 3.10. $\boldsymbol{T}_\Omega$ *is the smallest* Ω:-*subalgebra of* $\boldsymbol{T}_\Omega(V)$.

Proof. T_Ω is the smallest Ω-subalgebra of $\mathsf{T}_\Omega(V)$. □

Next we introduce a logic to specify properties of classes of type algebras.

Definition 3.11. The equational type logic of signature Ω and variables V has the following formulae on Ω and V:
atomic ET formulae:
(i) $t_1 \equiv t_2$ (equations),
(ii) $t_1 : t_2$ (type assignments) with $t_1, t_2 \in |\boldsymbol{T}_\Omega(V)|$;
ET formulae:
(iii) $\Gamma \to \alpha$
with α an atomic ET formula, called *conclusion*, and Γ a finite, possibly empty set of atomic ET formulae, called *assumption.*

Definition 3.12. An *ET presentation* is a triple $\langle \Omega, V, E \rangle$, where E is a set of ET formulae on Ω and V.

In the following we will often feel free to identify an ET presentation with its axioms E. In such a case, given V, we will assume that Ω is the smallest signature that is compatible with the axioms. The notions of *substitution*, *assignment*, and *evaluation* have the usual definitions. The *evaluation lemma* enables term evaluation to be determined by assignment.

Lemma 3.13. *For each Ω:-algebra A and assignment $\rho: V \to |A|$, a unique Ω:-morphism $\rho^{\#}: T_\Omega(V) \to A$ exists that extends ρ.*

Proof. The standard evaluation morphism $\rho^{\#}: T_\Omega(V) \to A$ uniquely extends ρ and vacuously respects the typing of $T_\Omega(V)$. □

Lemma 3.13 says that $T_\Omega(V)$ is free in Ω-Talg, the Ω-similarity class of type algebras; the unique extension of a substitution $\sigma: V \to |T_\Omega(V)|$ to the Ω:-morphism $\sigma^{\#}: T_\Omega(V) \to T_\Omega(V)$ is a special case. Initiality of T_Ω in Ω-Talg immediately follows.

Notation 3.14. The ET formula $\emptyset \to \alpha$ is identified with the atomic ET formula α. As a consequence of the evaluation lemma, we will often omit the superscript $^{\#}$ when evaluating terms according to a given assignment, as well as when referring to the substitution Ω:-morphism that is determined by a given substitution.

Definition 3.15. With given Ω and V, let t, u be terms of $T_\Omega(V)$, Γ an assumption, and $\Gamma \to \alpha$ an ET formula. The *satisfaction* relation "$\models$" between type algebras and ET formulae (extended to sets of them, ranged over by E) is defined as follows. Let ρ range over the evaluations $T_\Omega(V) \to A$, with A an Ω:-algebra.

(i) $A \models_\rho t \equiv u$ iff $\rho(t) = \rho(u)$;
(ii) $A \models_\rho t : u$ iff $\rho(t) :_A \rho(u)$;
(iii) $A \models_\rho \Gamma$ iff $A \models_\rho \alpha$ for all $\alpha \in \Gamma$;
(iv) $A \models_\rho \Gamma \to \alpha$ iff $A \models_\rho \Gamma$ implies $A \models_\rho \alpha$;
(v) $A \models \Gamma \to \alpha$ iff $A \models_\rho \Gamma \to \alpha$ $\forall \rho$;
(vi) $A \models E$ iff $A \models \Gamma \to \alpha$ $\forall \Gamma \to \alpha \in E$.

Definition 3.16. A *logical consequence* of a presentation $\langle \Omega, V, E \rangle$ is an ET formula ϕ on Ω and V such that $A \models \phi$ for every Ω:-algebra A that satisfies E; understanding Ω and V, this is also written $E \models \phi$ (read also *E logically implies ϕ*).

Definition 3.17. An *ET theory* is an ET presentation $\langle \Omega, V, E \rangle$ such that E is closed under logical consequence.

Definition 3.18. Talg(Ω, V, E) denotes the *class of models* of the ET presentation $\langle \Omega, V, E \rangle$, i.e. the class of type algebras that satisfy $\langle \Omega, V, E \rangle$. We will often omit V, i.e. assume that V is defined in the context: so Talg(Ω, E) is the class of Ω:-algebras that satisfy E. Talg$(\Omega, \emptyset)$ and Ω-Talg are synonyms.

The following notions enable an extension of the standard morphism theorem to type algebras (here stated without proof).

Definition 3.19. Let A be an Ω:-algebra, θ an Ω:-congruence on A, and $\phi: A \to B$ an Ω:-morphism; then:

(i) θ is a *strong Ω:-congruence on A* iff $:_\theta = :_A$,
(ii) ϕ is a *strong Ω:-morphism on A* iff $\phi(a) :_B \phi(a')$ implies $a :_A a'$.

Theorem 3.20. *Let $\phi : A \to B$ be an Ω:-morphism, and* Ker(ϕ) (*the* kernel *of* ϕ) *be the pair* $\langle \equiv_{\mathrm{Ker}(\phi)}, :_{\mathrm{Ker}(\phi)} \rangle$ *of binary relations on* $|A|$ *defined by*

$$a \equiv_{\mathrm{Ker}(\phi)} a' \text{ iff } \phi(a) = \phi(a'), \qquad a :_{\mathrm{Ker}(\phi)} a' \text{ iff } \phi(a) :_B \phi(a').$$

Then

(i) Ker(ϕ) *is an Ω:-congruence on A,*

(ii) Ker(ϕ) *is strong iff ϕ is strong,*

(iii) *there exists a unique Ω:-morphism $\phi_\kappa : A/\mathrm{Ker}(\phi) \to B$ such that $\phi = \phi_\kappa \circ \kappa_\phi$, where $\kappa_\phi : A \to A/\mathrm{Ker}(\phi)$ is the natural projection; moreover, ϕ_κ is injective and strong.*

Because of the conditional character of ET logic, one expects model classes to enjoy only quasi-varietal closure properties—in the general case. The next result (proof omitted) gives closure conditions for ET model classes. The straightforward extension of the standard definitions of homomorphic image and product algebras to type algebras is left to the reader.

Theorem 3.21. (i) *The class of models of every ET presentation is closed with respect to subalgebras and products,*

(ii) *the class of models of every ET presentation that consists of atomic formulae is closed with respect to homomorphic images, subalgebras and products,*

(iii) *the class of models of every ET presentation that has no equations in the assumptions is closed with respect to strong homomorphic images, subalgebras and products.*

4. The ET calculus

The ET *calculus* $\vdash$ is a binary relation between ET presentations and ET formulae that is constructed using two axiom schemas and eight inference rule schemas (collectively termed *rules* of the ET calculus, for short). These are presented in Table 1, where, understanding the signature Ω and variables V, we adopt the following notation.

Table 1. The rules of the ET calculus.

1.	$E \vdash \{\alpha\} \to \alpha$	Tautology
2.	If $E \vdash \Gamma \to \alpha$ then $E \vdash \Gamma \cup \{\beta\} \to \alpha$	Monotonicity
3.	$E \vdash t \equiv t$	Reflexivity
4.	If $E \vdash \Gamma \to t_1 \equiv t_2$ then $E \vdash \Gamma \to t_2 \equiv t_1$	Symmetry
5.	If $E \vdash \Gamma \to t_1 \equiv t_2$ and $E \vdash \Gamma \to t_2 \equiv t_3$ then $E \vdash \Gamma \to t_1 \equiv t_3$	Transitivity
6.	If $E \vdash \Gamma \to \alpha$ then $E \vdash \sigma(\Gamma) \to \sigma(\alpha)$	Substitution
7.	If $E \vdash \Gamma \to t_i \equiv u_i$ $(i = 1, \ldots, k)$ then $E \vdash \Gamma \to \omega(t_1, \ldots, t_k) \equiv \omega(u_1, \ldots, u_k)$	Replacement
8.	If $E \vdash \Gamma \cup \{\alpha\} \to \beta$ and $E \vdash \Gamma \to \alpha$ then $E \vdash \Gamma \to \beta$	Modus Ponens
9.	If $E \vdash \Gamma \to t_1 \equiv t_2$ and $E \vdash \Gamma \to t_1 : u$ then $E \vdash \Gamma \to t_2 : u$	Typing equals
10.	If $E \vdash \Gamma \to u_1 \equiv u_2$ and $E \vdash \Gamma \to t : u_1$ then $E \vdash \Gamma \to t : u_2$	Equating types

(i) t, u (possibly with subscripts) are terms,
(ii) α, β are atomic formulae,
(iii) Γ is an assumption,
(iv) ϕ is a formula,
(v) $\sigma: T_\Omega(V) \to T_\Omega(V)$ is a substitution, extended to formulae in the usual way,
(vi) ω is a k-ary operator.
We then define the following.

Definition 4.1. Relatively to an ET presentation $\langle \Omega, V, E \rangle$, the set of ET formulae ϕ derivable from $\langle \Omega, V, E \rangle$, we write $E \vdash \phi$, is the smallest set that includes E and is closed with respect to the rules of the ET calculus (see Table 1).

Proposition 4.2. *The ET calculus is* sound: *if the type algebra A satisfies the presentation E, then it satisfies any formula derivable from E; in symbols*: $(A \models E \wedge E \vdash \phi) \Rightarrow A \models \phi$.

Proof. We show the soundness of rule 6 (soundness of the other rules is obvious). If ρ is an evaluation such that $A \models_\rho \sigma(\Gamma)$, then $\rho \circ \sigma : T_\Omega(V) \to A$ is an evaluation such that $A \models_{\rho\circ\sigma} \Gamma$. Then, by hypothesis, $A \models_{\rho\circ\sigma} \alpha$, i.e. $A \models_\rho \sigma(\alpha)$. □

Definition 4.3. Let $\langle \Omega, V, E \rangle$ be an ET presentation. E_Γ are the members of an assumption-indexed *family of syntactic Ω:-congruences* on $T_\Omega(V)$, where we define:
(i) $t \equiv_{E_\Gamma} u$ iff $E \vdash \Gamma \to t \equiv u$,
(ii) $t :_{E_\Gamma} u$ iff $E \vdash \Gamma \to t : u$.

Notation 4.4. If $\theta = \langle \equiv_\theta, :_\theta \rangle$ is an Ω:-congruence on $T_\Omega(V)$, we write $t \equiv u \in \theta$ for $t \equiv_\theta u$ and $t : u \in \theta$ for $t :_\theta u$.

Lemma 4.5. *For given Ω and V, E_Γ is an Ω:-congruence on $T_\Omega(V)$ for all E and Γ, that has the following properties*:
(a) *if $\Gamma \to \alpha \in E$, then $\alpha \in E_\Gamma$*;
(b) *if $\alpha \in \Gamma$, then $\alpha \in E_\Gamma$*;
(c) *if $\alpha \in E_\Gamma$, then $\alpha \in E_{\Gamma\cup\{\beta\}}$ for all atomic formulae β*;
(d) *if $\alpha \in E_\Gamma$, then $\sigma(\alpha) \in E_{\sigma(\Gamma)}$ for all substitutions $\sigma: T_\Omega(V) \to T_\Omega(V)$*;
(e) *if $\alpha \in E_{\Gamma\cup\{\beta\}}$ and $\beta \in E_\Gamma$, then $\alpha \in E_\Gamma$*.

Proof. By inspection on the appropriate rules of the ET calculus with respect to Definition 3.5 (whose condition (iv) is fulfilled because the typing of $T_\Omega(V)$ is empty—see Definition 3.8) and to the properties stated above. □

Proposition 4.6. *Every quotient $T_\Omega(V)/E_\Gamma$ satisfies the conditions*:
(i) $T_\Omega(V)/E_\Gamma \models E$,
(ii) $T_\Omega(V)/E_\Gamma \models \Gamma \to \alpha \Leftrightarrow E \vdash \Gamma \to \alpha$.

Proof. Let T/Γ abbreviate $T_\Omega(V)/E_\Gamma$ in this proof.

(i) Let $\Gamma' \to \alpha \in E$. We have to show that $T/\Gamma \models_\pi \alpha$ for all $\pi: T_\Omega(V) \to T/\Gamma$ such that $T/\Gamma \models_\pi \Gamma'$. To this end, given any such π, we introduce a function $\rho: V \to |T_\Omega(V)|$ that meets the condition $[\rho(x)]_{E_\Gamma} = \pi(x)\ \forall x \in V$. It is then easy to verify by induction that $[\rho(t)]_{E_\Gamma} = \pi(t)$ for all $t \in |T_\Omega(V)|$. This entails that $[\rho(t)]_{E_\Gamma} = [\rho(u)]_{E_\Gamma}$ iff $\pi(t) = \pi(u)$, and also that $[\rho(t)]_{E_\Gamma} :_{T/\Gamma} [\rho(u)]_{E_\Gamma}$ iff $\pi(t) :_{T/\Gamma} \pi(u)$, whence by Definition 3.6:

$$\rho(t) \equiv_{E_\Gamma} \rho(u) \text{ iff } \pi(t) = \pi(u) \quad \text{and} \quad \rho(t) :_{E_\Gamma} \rho(u) \text{ iff } \pi(t) :_{T/\Gamma} \pi(u). \quad (*)$$

It is then sufficient to show that $\rho(\alpha) \in E_\Gamma$; this has the following proof: $\alpha \in E_\Gamma$ by Lemma 4.5(a), whence $\rho(\alpha) \in E_{\rho(\Gamma')}$ by Lemma 4.5(d), whence $\rho(\alpha) \in E_{\Gamma \cup \rho(\Gamma')}$ by Lemma 4.5(c), and since $\rho(\gamma) \in E_\Gamma\ \forall \gamma \in \Gamma'$ by the hypothesis $T/\Gamma \models_\pi \Gamma'$ and fact (*) above, we conclude that $\rho(\alpha) \in E_\Gamma$ by Lemma 4.5(e).

(ii) ($\Leftarrow$) follows from (i) and soundness (Proposition 4.2).

($\Rightarrow$) Let $\pi: V \to |T/\Gamma|$ be the assignment that maps every variable x to its equivalence class $[x]_{E_\Gamma}$: by the evaluation Lemma (3.13) $\pi(t) = [t]_{E_\Gamma}$ for every term t; then, for every atomic formula β, $T/\Gamma \models_\pi \beta$ iff $\beta \in E_\Gamma$, whence $T/\Gamma \models_\pi \Gamma$ by Lemma 4.5(b). This and the hypothesis entail $T/\Gamma \models_\pi \alpha$, hence $\alpha \in E_\Gamma$. Then $E \vdash \Gamma \to \alpha$ by Definition 4.3. □

Theorem 4.7. *The ET calculus is* complete: *if the formula ϕ is a logical consequence of the presentation $\langle \Omega, V, E \rangle$, then it is derivable from $\langle \Omega, V, E \rangle$; together with the soundness (Proposition 4.2), this is formulated as: $E \vdash \phi \Leftrightarrow E \models \phi$.*

Proof. By Proposition 4.6(i) $E \models \Gamma \to \alpha \Rightarrow T_\Omega(V)/E_\Gamma \models \Gamma \to \alpha$, whence $E \vdash \Gamma \to \alpha$ by Proposition 4.6(ii). □

Notation 4.8. We let E denote also the syntactic Ω:-congruence $E_\emptyset$, when this gives rise to no ambiguity.

Theorem 4.9. *If A is a type algebra that satisfies the ET presentation $\langle \Omega, V, E \rangle$, then for every assignment $\rho: V \to |A|$ there exists a unique Ω:-morphism $\rho^\#_E: T_\Omega(V)/E \to A$ that extends ρ.*

Proof. The Ω-morphism $\rho^\#_E: T_\Omega(V)/\equiv_E \to A$ defined by $\rho^\#_E([t]_E) = \rho^\#(t)$ uniquely extends ρ, by a classical result. Moreover, letting $T/$abbreviate $T_\Omega(V)/E$, we show that $\rho^\#_E$ respects $:_{T/}$

$$\begin{aligned}
[t]_E :_{T/} [u]_E &\Rightarrow t :_E u && \text{by Definition 3.6}\\
&\Rightarrow E \vdash t : u && \text{by Definition 4.3}\\
&\Rightarrow A \models t : u && \text{by Proposition 4.2 and the hypothesis}\\
&\Rightarrow \rho^\#(t) :_A \rho^\#(u) && \text{by Lemma 3.13}\\
&\Rightarrow \rho^\#_E([t]_E) :_A \rho^\#_E([u]_E) && \text{by the Definition of } \rho^\#_E. \quad \square
\end{aligned}$$

Corollary 4.10. *T_Ω/E is* initial *in the class of Ω:-algebras that satisfy E.*

5. ET specification examples

The ET specifications given in Table 2 illustrate solutions to the problem examples proposed in Section 2 (substantially larger ET specifications are proposed in [26]). We informally introduce the syntax of the simple specification language adopted.

Table 2. ET specification examples

Example 5.1

spec NAT **is**
0 : nat;
n : nat → succ(n) : nat;
n : nat → pred(succ(n)) ≡ n
endspec

Example 5.2

spec NATERR **is** NAT;
pred(0) : error(nat);
x : error(nat) → pred(x) : error(nat);
x : error(nat) → succ(x) : error(nat)
endspec

Example 5.3

spec INTNATERR **is** NATERR;
n : nat → n : int;
n : int → pred(n) : int;
n : int → pred(succ(n)) ≡ n;
n : int → succ(pred(n)) ≡ n
endspec

Example 5.4

spec IDENTITY **is**
d, c : type → fun(d, c) : type;
d, c : type, x : d, f : fun(d, c) → apply(f, x) : c;
t : type → id t : fun(t, t);
t : type, x : t → apply(id t, x) ≡ x
endspec

Example 5.5

spec NATINTERVAL **is** NAT;
n : nat → n : interval(0, n);
n : nat, m : interval(0, pred(n)) → m : interval(0, n)
endspec

Basically, an ET specification is a named, finite ET presentation. We convene that a countable set of variables V is available, whose members will be the alphanumeric strings in this ***boldface italics*** font. Identifiers that occur in some formula of E and are not in V form the (one-sorted) signature of the presentation, with the arity of each operator uniquely determined by the number of arguments it takes in every term of ET formulae in E where it occurs. For the sake of brevity, we write e.g. "$s, t : u$" instead of the more customary "$s : u$, $t : u$". Also, the syntax allows prefix notation for unary operators.

The basic syntax is enriched by two specification-building constructs, which are commonly found in any algebraic specification language of practical interest: (1) putting together specification modules, by means of the ";" operator, (2) invoking modules by reference to their names. The design of an ET specification language is not in the scope of the present paper; nonetheless, the formal developments and results presented in Section 6 should suffice to justify this additional syntax.

Example 5.1 (*Partiality*) We observe that no type can be assigned to "pred(0)" by the axioms, nor is "pred(0)" a type either, so we view this term as representing an underdefined element; it is easy to see that, under initial semantics, "pred(t)" denotes an underdefined element if term "t" does so.

This example offers the opportunity to address a problem that is known to arise in equational order-sorted specification, where exceptions can be specified by distinguishing the subsort "pos" of the positive (i.e. non-zero) natural numbers from the sort "nat" of all natural numbers. The predecessor operation would only be defined on positive numbers, but its application may yield zero, hence the signature declarations: "post < nat", "zero : nat", "succ : nat → pos", "pred : pos → nat". This entails syntactic restrictions, e.g. the equation "pred(pred(succ(succ(zero)))) ≡ zero" is not allowed, since the left-hand side term is ill-formed in the given signature. An operational way out is discussed in [10], that introduces auxiliary operations, termed *retracts*. No auxiliary device is necessary in ET logic, since type assignment is available for classifying exceptions (if desired). In particular, the identity above is derivable, e.g. by the following proof, where we adopt the following conventions:

(1) derivation steps are numbered for further reference,
(2) in each derivation step:
  (2.1) reference to the ET rule of interest is made by the subscript attached to the derivation symbol $\vdash$,
  (2.2) reference to NAT axioms (in the form "A*n*", denoting the *n*th axiom in the NAT specification), and to formulae derived in previous derivation steps, is made in brackets at the right-hand side.

NAT

(1)	$\vdash_6$ zero : nat → succ(zero) : nat	[A2]
(2)	$\vdash_8$ succ(zero) : nat	[1, A1]
(3)	$\vdash_6$ succ(zero) : nat → pred(succ(succ(zero))) ≡ succ(zero)	[A3]
(4)	$\vdash_8$ pred(succ(succ(zero))) ≡ succ(zero)	[3, 2]
(5)	$\vdash_7$ pred(pred(succ(succ(zero)))) ≡ pred(succ(zero))	[4]
(6)	$\vdash_6$ zero : nat → pred(succ(zero)) ≡ zero	[A3]
(7)	$\vdash_8$ pred(succ(zero)) ≡ zero	[6, A1]
(8)	$\vdash_5$ pred(pred(succ(succ(zero)))) ≡ zero	[5, 7].

Example 5.2 (*Exception handling*). The type denoted by "error(nat)" classifies elements that are underdefined in NAT: not all of such elements (e.g. "pred(nat)" and "succ(nat)" still denote underdefined elements), since this is not required. Error propagation is specified explicitly (second and third axiom in the example). Error classification is made relative to "nat" by term construction.

Example 5.3 (*Extension*). The notion of exception recovery by extension is illustrated with the refinement of Example 5.2 by Example 5.3, where we observe that "pred(0)" has two types: it is an error term when interpreted within the "small horizon" of the natural numbers, but it is a correct term within the larger one of

the integers. Enlightened by the wider significance of the latter, one might like to forget about the previous error classification, and retain only the normal classifications, according to the so extended "norm". Semantics for such "hiding" facilities are usually found in the algebraic notion of *reduction*, where the hidden operations are forgotten. Now, reduction in many-sorted algebras forgets operations, but not values unless their whole carrier is forgotten (by hiding their sort). In particular, reduction in classical one-sorted algebras never forgets any values. This form of reduction is available also for type algebras, but is not the only one available. Several alternatives exist, characterized by increasing forgetting power: the formal treatment of reduction and extension for type algebras in addressed in Section 6.

Example 5.4 (*Type polymorphism*). Here the "type" constant is meant to stand for some kind of "type of all types". An initial definition for it is to be provided in the context where the module is made use of. For instance, when using the module in the context of a functional programming language, the context would define "type" as the (higher-order) type of the basic types of that language. This example demonstrates that having no fundamental difference between types and elements makes it possible to amend the expression of type polymorphism and dependent types to a common root, that is functional dependence.

Example 5.5 (*Dependent types*). We specify the intervals $[0, n]$ on the natural numbers by corresponding type assignments, using the operations available from the NAT specification above. A direct specification (i.e. with no reference to NAT) is also feasible; we leave it to the reader.

6. Formal notions of reduction and extension

Formal notions of reduction and extension, as conceptual and formal tools, prove essential to software engineering applications, in that they enable the introduction of specification structuring facilities [8]. The formal treatment of these notions in ET logic starts from a standard definition and a straightforward extension of a standard algebraic notion.

Definition 6.1. A *signature morphism* $\tau: \Omega \to \Omega'$ is a map that preserves arities.

Definition 6.2. If $\tau: \Omega \to \Omega'$ is a signature morphism and A is an Ω'-algebra, the *τ-reduct* of A, written $A{\downarrow}\tau$, is the Ω-algebra that has the same carrier and typing as A, and the operations $\omega^{A{\downarrow}\tau} = (\tau\omega)^A$ for all ω in Ω.

Definition 6.2 can be related to the standard notion of reduct algebra as follows: take $\tau: \Omega \to \Omega'$ to be an inclusion map, then this special case of τ-reduct coincides with the standard notion of one-sorted Ω-reduct. We turn $\_{\downarrow}\tau$ into a functor by

letting it map also Ω':-morphisms into Ω:-morphisms in the usual way: if $\tau: \Omega \to \Omega'$ is an ET signature morphism, A and B Ω':-algebras, and $\phi: A \to B$ an Ω':-morphism, the Ω:-morphism $\phi\downarrow\tau: A\downarrow\tau \to B\downarrow\tau$ is defined by $\phi\downarrow\tau(a) = \phi(a)\ \forall a \in |A\downarrow\tau|$. The relative weakness of the notion of reduction as from Definition 6.2, in comparison with reduction in many-sorted algebra, is immediately seen; if $\Sigma \subseteq \Sigma'$ are many-sorted signatures, with $S \subseteq S'$ the respective sort sets, then the Σ-reduct of a Σ'-algebra forgets exactly those values that have no sort in S; on the contrary, no values are forgotten by $\_\downarrow\tau$, but only the operations that have no denotation in Ω. The same problem was solved by Mosses [22] by introducing a "more forgetful" reduction. The essence of Mosses' idea is that of finding a substructure of the "weak" reduct where the "undesired" values are forgotten. This idea can be exploited for type algebras in a more general, hierarchical way, by using the subalgebra concept. We take the strongest reduct to be the smallest subalgebra of the weakest reduct.

Definition 6.3. Let $\tau: \Omega \to \Omega'$ be a signature morphism and A an Ω':-algebra; then the *Ω_0-generated τ-reduct* of A, written $A\downarrow^0\tau$, is the Ω:-subalgebra of $A\downarrow\tau$ that has carrier $|A\downarrow^0\tau| = \{t^{A\downarrow\tau} \mid t \in |T_\Omega|\}$.

With this reduction, exactly those elements of the carrier are forgotten that have no denotation in the reduct's signature. This reduct is the base of an ordinal hierarchy of weaker reducts, which naturally arises, by weakening the requirement that characterizes the base, in a "bidirectional", inductive fashion—consistently with the binary nature of the typing relation.

Definition 6.4. Let τ, A be as in Definition 6.3, and ν a pair of ordinals α, β, written $\nu = \alpha:\beta$. The *Ω_ν-generated τ-reduct* of A, written $A\downarrow^\nu\tau$, is inductively defined as the least Ω:-algebra B that satisfies the following requirements:

(i) $B \subseteq A\downarrow\tau$,

(ii) $\xi < \alpha \Rightarrow (A\downarrow^{\xi:\beta}\tau \subseteq B \,\&\, \forall a \in |A|\,.\,((a :_A b \,\&\, b \in |A\downarrow^{\xi:\beta}\tau|) \Rightarrow a \in |B|))$,

(iii) $\zeta < \beta \Rightarrow (A\downarrow^{\alpha:\zeta}\tau \subseteq B \,\&\, \forall b \in |A|\,.\,((a :_A b \,\&\, a \in |A\downarrow^{\alpha:\zeta}\tau|) \Rightarrow b \in |B|))$.

Clearly, this definition entails $A\downarrow^{0:0}\tau = A\downarrow^0\tau$. Moreover, pairs of ordinals naturally form a lattice: we extend this with a top element $\mathbf{1}$, let also $\mathbf{0} = 0{:}0$ denote the bottom, and let ν, μ range over this complete lattice. In correspondence with this completion, we get from Definition 6.4 a complete sublattice of the substructure lattice of $A\downarrow\tau$ by adding this weakest reduct as top of the sublattice: $A\downarrow^{\mathbf{1}}\tau =_{\text{def}} A\downarrow\tau$. The $\Omega_{1:0}$-generated reduct generalizes many-sorted reduct in the following way. In $A\downarrow^{1:0}\tau$ only those values are retained that have denotation or denotable type in Ω, or result from application of some of the retained operations to a retained value, and this clearly corresponds to the carrier-forgetting power of many-sorted reduction.

We turn every reduction in the lattice into a functor in the usual way: if $\tau: \Omega \to \Omega'$ is a signature morphism and $\phi: A \to B$ an Ω':-morphism, the Ω:-morphism $\phi\downarrow^\nu\tau: A\downarrow^\nu\tau \to B\downarrow^\nu\tau$ is defined by $\phi\downarrow^\nu\tau(a) = \phi(a)\ \forall a \in |A\downarrow^\nu\tau|$. The easy check that

$\phi\downarrow^{\nu}\tau$ is an Ω:-morphism is left to the reader. Definition 6.4 transfers the nice idea that is embedded in Mosses'"more forgetful" functor [22] to our framework: further comparison will proceed along the analysis of our notions of reduction. The next result tells us what happens when a reduction is "followed" by a stronger one (cf. Lemma 5 in [22]). The preliminary proposition states special cases of the main theorem, but these are needed for the proof of the more general case.

Lemma 6.5. *Let $A \subseteq B$ in Ω'-Talg, $\tau: \Omega \to \Omega'$ a signature morphism. $\nu \leqslant \mu \Rightarrow A\downarrow^{\nu}\tau \subseteq B\downarrow^{\mu}\tau$.*

Proof. Immediate. □

Proposition 6.6. *Let $\theta: \Omega \to \Omega'$, $\tau: \Omega' \to \Omega''$ be signature morphisms, and $A \in \Omega''$-Talg; then*

(i) $A\downarrow\tau\downarrow\theta = A\downarrow\tau \circ \theta$,

(ii) $A\downarrow\tau\downarrow^{\nu}\theta = A\downarrow^{\nu}\tau \circ \theta$,

(iii) $\nu \leqslant \mu \Rightarrow A\downarrow\tau\downarrow^{\nu}\theta \subseteq A\downarrow^{\mu}\tau\downarrow\theta$.

Proof. (i) Obvious.

(ii) The case $\nu = \mathbf{1}$ coincides with (i). For $\nu = \alpha{:}\beta$, the proof is by induction on ν. The basis ($\nu = \mathbf{0}$) follows from (i), since $A\downarrow\tau\downarrow^{\mathbf{0}}\theta$ is the minimal Ω:-subalgebra of $A\downarrow\tau\downarrow\theta$ and $A\downarrow^{\mathbf{0}}\tau \circ \theta$ is the minimal Ω:-subalgebra of $A\downarrow\tau \circ \theta$.

Inductive step: by Definition 6.4, $A\downarrow\tau\downarrow^{\alpha:\beta}\theta$ is the least Ω:-subalgebra B of $A\downarrow\tau\downarrow\theta$ that satisfies the requirements that are obtained from (ii) and (iii) of that definition by substituting θ for τ and $A\downarrow\tau$ for A; the induction hypothesis entails that $A\downarrow^{\alpha:\beta}\tau \circ \theta$ is the least Ω:-subalgebra B of $A\downarrow\tau \circ \theta$ that satisfies the same requirements; since $A\downarrow\tau \circ \theta = A\downarrow\tau\downarrow\theta$ by (i), those two least subalgebras coincide.

(iii) The case $\mu = \mathbf{1}$ is immediate. The rest of the proof is by double induction on μ and ν. External basis ($\mu = \mathbf{0}$): assume $\nu = \mathbf{0}$. $A\downarrow\tau\downarrow^{\mathbf{0}}\theta$ is the minimal Ω:-subalgebra of $A\downarrow\tau\downarrow\theta$. Moreover, $A\downarrow^{\mathbf{0}}\tau \subseteq A\downarrow\tau \Rightarrow A\downarrow^{\mathbf{0}}\tau\downarrow\theta \subseteq A\downarrow\tau\downarrow\theta$ by Lemma 6.5. So, $A\downarrow\tau\downarrow^{\mathbf{0}}\theta$ is the minimal Ω:-subalgebra of $A\downarrow^{\mathbf{0}}\tau\downarrow\theta$ as well.

External inductive step: Internal basis ($\nu = \mathbf{0}$): similar to the external basis, but considering $A\downarrow^{\mu}\tau\downarrow\theta \subseteq A\downarrow\tau\downarrow\theta$. Internal inductive step: assume $\mathbf{0} < \nu = \alpha{:}\beta \leqslant \mu$. It is sufficient to show that $A\downarrow^{\mu}\tau\downarrow\theta$ is an Ω:-subalgebra B of $A\downarrow\tau\downarrow\theta$ that satisfies the requirements in the definition of $A\downarrow\tau\downarrow^{\xi:\beta}\theta$, which are obtained from (ii) and (iii) of Definition 6.4 by substituting θ for τ and $A\downarrow\tau$ for A; in particular, concerning requirement (ii), we must show that

$$\xi < \alpha \;\Rightarrow\; (A\downarrow\tau\downarrow^{\xi:\beta}\theta \subseteq A\downarrow^{\mu}\tau\downarrow\theta \;\&\; \forall a \in |A\downarrow\tau|\,.\,((a :_A b \;\&\; b \in |A\downarrow\tau\downarrow^{\xi:\beta}\theta|) \Rightarrow a \in |A\downarrow^{\mu}\tau\downarrow\theta|)).$$

By internal induction hypothesis we get immediately $A\downarrow\tau\downarrow^{\xi:\beta}\theta \subseteq A\downarrow^{\mu}\tau\downarrow\theta$. The remaining condition concerns only the carriers, viz. we assume $a \in |A\downarrow\tau| = |A|$ and $b \in |A\downarrow\tau\downarrow^{\xi:\beta}\theta|$ such that $a :_A b$, and we have to show that $a \in |A\downarrow^{\mu}\tau\downarrow\theta|$; it is sufficient

to show $a \in |A\downarrow^{\nu}\tau|$, since $|A\downarrow^{\nu}\tau| \subseteq |A\downarrow^{\mu}\tau|$ follows from $\nu \leqslant \mu$, and $|A\downarrow^{\mu}\tau| = |A\downarrow^{\mu}\tau\downarrow\theta|$. Now, consider the definition of $A\downarrow^{\nu}\tau$. We have by assumption that $a \in |A|$, $a :_A b$, and $b \in |A\downarrow\tau\downarrow^{\xi:\beta}\theta|$; the last fact entaiis $b \in |A\downarrow^{\xi:\beta}\tau|$ (because $|A\downarrow\tau\downarrow^{\xi:\beta}\theta| \subseteq |A\downarrow^{\xi:\beta}\tau\downarrow\theta|$ by induction hypothesis, and $|A\downarrow^{\xi:\beta}\tau\downarrow\theta| = |A\downarrow^{\xi:\beta}\tau|$): since $A\downarrow^{\nu}\tau$ must satisfy requirement (ii) of Definition 6.4, we can conclude that $a \in |A\downarrow^{\nu}\tau|$. It is similarly shown that

$$\zeta < \beta \Rightarrow (A\downarrow\tau\downarrow^{\alpha:\zeta}\theta \subseteq A\downarrow^{\mu}\tau\downarrow\theta \;\&\; \forall b \in |A\downarrow\tau| \,.\, ((a :_A b \;\&\; a \in |A\downarrow\tau\downarrow^{\alpha:\zeta}\theta|) \Rightarrow b \in |A\downarrow^{\mu}\tau\downarrow\theta|)). \quad \square$$

Theorem 6.7. *Let $\theta : \Omega \to \Omega'$, $\tau : \Omega' \to \Omega''$ be signature morphisms, and $A \in \Omega''$-*Talg*; then*

$$\nu \leqslant \mu \Rightarrow A\downarrow^{\mu}\tau\downarrow^{\nu}\theta = A\downarrow^{\nu}\tau \circ \theta = A\downarrow\tau\downarrow^{\nu}\theta.$$

Proof. The second identity is just Proposition 6.6(ii). We show $A\downarrow\tau\downarrow^{\nu}\theta \subseteq A\downarrow^{\mu}\tau\downarrow^{\nu}\theta$ by induction on ν, assuming $\nu = \alpha:\beta \leqslant \mu$ (the converse statement follows from Lemma 6.5, and the case $\nu = \mu = \mathbf{1}$ is covered by Proposition 6.6(i).

Basis ($\nu = \mathbf{0}$): $A\downarrow\tau\downarrow^{\mathbf{0}}\theta$ is the minimal Ω:-subalgebra of $A\downarrow\tau\downarrow\theta$. Lemma 6.5 gives $A\downarrow^{\mu}\tau\downarrow^{\mathbf{0}}\theta \subseteq A\downarrow\tau\downarrow^{\mathbf{0}}\theta \subseteq A\downarrow\tau\downarrow\theta$. Then, $A\downarrow\tau\downarrow^{\mathbf{0}}\theta$ is the minimal Ω:-subalgebra of $A\downarrow^{\mu}\tau\downarrow^{\mathbf{0}}\theta$ as well.

Inductive step: it is sufficient to show that $A\downarrow^{\mu}\tau\downarrow^{\nu}\theta$ is an Ω:-subalgebra $\boldsymbol{B}$ of $A\downarrow\tau\downarrow\theta$ that satisfies the requirements in the definition of $A\downarrow\tau\downarrow^{\nu}\theta$, which are obtained from (ii) and (iii) of Definition 6.4 by substituting θ for τ and $A\downarrow\tau$ for A; in particular, concerning requirement (ii), we show that

$$\xi < \alpha \Rightarrow (A\downarrow\tau\downarrow^{\xi:\beta}\theta \subseteq A\downarrow^{\mu}\tau\downarrow^{\nu}\theta \;\&\; \forall a \in |A\downarrow\tau| \,. ((a :_A b \;\&\; b \in |A\downarrow\tau\downarrow^{\xi:\beta}\theta|) \Rightarrow a \in |A\downarrow^{\mu}\tau\downarrow^{\nu}\theta|)).$$

By induction hypothesis and Lemma 6.5 we get immediately $A\downarrow\tau\downarrow^{\xi:\beta}\theta = A\downarrow^{\mu}\tau\downarrow^{\xi:\beta}\theta \subseteq A\downarrow^{\mu}\tau\downarrow^{\nu}\theta$. About the remaining condition, we assume

$$a \in |A\downarrow\tau| = |A| \quad \text{and} \quad b \in |A\downarrow\tau\downarrow^{\xi:\beta}\theta| = |A\downarrow^{\mu}\tau\downarrow^{\xi:\beta}\theta| \quad \text{(induction hypothesis)}$$

such that $a :_A b$ and show that $a \in |A\downarrow^{\mu}\tau\downarrow^{\nu}\theta|$. We show that $a \in |A\downarrow^{\nu}\tau|$, which implies $a \in |A\downarrow^{\mu}\tau|$; this and $b \in |A\downarrow^{\mu}\tau\downarrow^{\xi:\beta}\theta|$ entail $a \in |A\downarrow^{\mu}\tau\downarrow^{\xi+1:\beta}\theta|$ (by Definition 6.4), with $\xi+1:\beta \leqslant \nu \leqslant \mu$, and $A\downarrow^{\mu}\tau\downarrow^{\xi+1:\beta}\theta \subseteq A\downarrow^{\mu}\tau\downarrow^{\nu}\theta$ by Lemma 6.5, whence $a \in |A\downarrow^{\mu}\tau\downarrow^{\nu}\theta|$ will follow. So, let us show $a \in |A\downarrow^{\nu}\tau|$. We have by assumption that $a \in |A|$, $a :_A b$, and $b \in |A\downarrow\tau\downarrow^{\xi:\beta}\theta|$. Since $A\downarrow\tau\downarrow^{\xi:\beta}\theta \subseteq A\downarrow^{\xi:\beta}\tau\downarrow\theta$ by Proposition 6.6(iii), and $|A\downarrow^{\xi:\beta}\tau\downarrow\theta| = |A\downarrow^{\xi:\beta}\tau|$, we have $b \in |A\downarrow^{\xi:\beta}\tau|$. But $A\downarrow^{\nu}\tau$ must satisfy requirement (ii) of Definition 6.4, so $a \in |A\downarrow^{\nu}\tau|$. It is similarly shown that

$$\zeta < \beta \Rightarrow (A\downarrow\tau\downarrow^{\alpha:\zeta}\theta \subseteq A\downarrow^{\mu}\tau\downarrow^{\nu}\theta \;\&\; \forall b \in |A\downarrow\tau| \,. ((a :_A b \;\&\; a \in |A\downarrow\tau\downarrow^{\alpha:\zeta}\theta|) \Rightarrow b \in |A\downarrow^{\mu}\tau\downarrow^{\nu}\theta|)). \quad \square$$

The next definition introduces a useful special case of the notion of signature morphism. The following notational convention is adopted: if $\tau : \Omega \to \Omega'$ is a signature morphism and $\langle \Omega, V, E \rangle$ an ET presentation, we let τE denote the set of ET

formulae on Ω' and V that is obtained from E by replacing, in every formula of E, every occurrence of each operator $\omega \in \Omega$ with $\tau\omega$.

Definition 6.8. If $\langle \Omega, V, E\rangle$, $\langle \Omega', V, E'\rangle$ are ET theories, an ET *theory morphism* $\vartheta : \langle \Omega, V, E\rangle \to \langle \Omega', V, E'\rangle$ is a signature morphism $\vartheta : \Omega \to \Omega'$ such that $\vartheta E \subseteq E'$.

As a matter of notation, if $\langle \Omega, V, E\rangle$, $\langle \Omega', V, E'\rangle$ are ET presentations, we write $\vartheta : \langle \Omega, E\rangle \to \langle \Omega', E'\rangle$, or even $\vartheta : E \to E'$, understanding that $E' \models \vartheta E$, to denote the ET theory morphism $\vartheta : \langle \Omega, V, \Theta\rangle \to \langle \Omega', V, \Theta'\rangle$, where $\Theta(\Theta')$ is the closure of $E(E')$ under logical consequence, with the given $\Omega(\Omega')$ and V. Clearly, for every ET theory morphism $\vartheta : E \to E'$, $\_\downarrow\vartheta$ maps models of E' to models of E. In fact, *every* reduction in the lattice preserves satisfaction, as the next proposition shows (cf. Proposition 4 in [22]).

Proposition 6.9. *If $\vartheta : \langle \Omega, E\rangle \to \langle \Omega', E'\rangle$ is an ET theory morphism and $A \models E'$, then $A\downarrow^{\nu}\vartheta \models E$ for all ν.*

Proof. $A\downarrow\vartheta \models E$ by Definitions 6.1 and 6.8, $A\downarrow^{\nu}\vartheta$ is an Ω:-subalgebra of $A\downarrow\vartheta$, and the universally quantified ET formulae valid in $A\downarrow\vartheta$ hold a fortiori in the Ω:-subalgebras of $A\downarrow\vartheta$. □

A few concepts and related notation are needed, to construct the counterpart of reduction (relative to a theory morphism).

Notation 6.10. Let Ω be an ET signature, C a set, and A a type algebra: $\Omega(C)$ denotes the signature that extends Ω by adding the elements of C as constants; $\Omega(A)$ is shorthand for $\Omega(|A|)$ (please note: A is any type algebra, and no relationship is required between A and Ω). If $\tau : \Omega \to \Omega'$ is an ET signature morphism, then $\tau_A : \Omega(A) \to \Omega'(A)$ extends τ by letting $\tau_A(a) = a$ $\forall a \in |A|$.

Definition 6.11. If A is an Ω:-algebra, the *ET diagram of A* is the set $\delta(A)$ of atomic ET formulae on $\Omega(A)$ that is defined by:

$$\begin{aligned}\delta(A) = &\{a \equiv t \mid a \in |A|,\ t \in |T_{\Omega(A)}|,\ a = t^A\}\\ &\cup \{a{:}t \mid a \in |A|,\ t \in |T_{\Omega(A)}|,\ a :_A t^A\}\\ &\cup \{t{:}a \mid a \in |A|,\ t \in |T_{\Omega(A)}|,\ t^A :_A a\}.\end{aligned}$$

Definition 6.12. Let $A \in \mathrm{Talg}(\Omega, E)$ and $\vartheta : \langle \Omega, E\rangle \to \langle \Omega', E'\rangle$ be an ET theory morphism: $\approx(\vartheta, A)$ is the $\Omega'(A)$:-congruence on $T_{\Omega'(A)}$ such that $T_{\Omega'(A)}/\approx(\vartheta, A)$ is initial in $\mathrm{Talg}(\Omega'(A), E' \cup \vartheta_A\delta(A))$.

Proposition 6.13. *Let A and ϑ be as in Definition* 6.12, *and* $\approx(A, \vartheta) =_{\text{def}}$ *the restriction of* $\approx(\vartheta, A)$ *to* $|A|^2$; *then,* $\approx(A, \vartheta)$ *is an* Ω:-*congruence on* A

Proof. Immediate. □

We are ready now to propose the following formalization of the concept of ϑ-extension.

Definition 6.14. Let $A \in \mathrm{Talg}(\Omega, E)$, $A' \in \mathrm{Talg}(\Omega', E')$, and $\vartheta : \langle \Omega, E \rangle \to \langle \Omega', E' \rangle$ be an ET theory morphism; then A' is a *ϑ-extension* of A iff there exists an Ω:-morphism $\psi : A \to A'\!\downarrow\vartheta$ such that $\mathrm{Ker}(\psi) = \approx(A, \vartheta)$.

The first step in the analysis of the relationship between ϑ-extension and ϑ-reduction is a simple fact. The second step is the search for minimal ϑ-extensions; the definition below instantiates a standard categorical notion.

Proposition 6.15. *If $\vartheta : E \to E'$ is an ET theory morphism and $A' \models E'$, then A' is a ϑ-extension of every ϑ-reduct $A'\!\downarrow^{v}\vartheta$.*

Proof. $A'\!\downarrow^{v}\vartheta \models E$ by Proposition 6.9, and the inclusion morphism $\psi : A'\!\downarrow^{v}\vartheta \to A'\!\downarrow\vartheta$ satisfies the condition in Definition 6.14. □

Definition 6.16. If $\vartheta : \langle \Omega, E \rangle \to \langle \Omega', E' \rangle$ is an ET theory morphism, $A \in \mathrm{Talg}(\Omega, E)$, $A' \in \mathrm{Talg}(\Omega', E')$, then A' is a *ϑ-free extension* of A iff there exists an Ω:-morphism $\eta_A : A \to A'\!\downarrow\vartheta$ such that, for every $B \in \mathrm{Talg}(\Omega', E')$ and Ω:-morphism $\phi : A \to B\!\downarrow\vartheta$, a unique Ω':-morphism $\phi^{\#} : A' \to B$ exists such that $\phi^{\#}\!\downarrow\vartheta \circ \eta_A = \phi$.

Proposition 6.17. *Let $\vartheta : \langle \Omega, E \rangle \to \langle \Omega', E' \rangle$ be an ET theory morphism and $A \in \mathrm{Talg}(\Omega, E)$: if a ϑ-free extension of A exists it is unique up to isomorphism.*

Proof. Categorical routine. □

A construction of ϑ-free extensions is exhibited, that generalizes the similar construction for the many-sorted equational case (see e.g. [9]). The following convention is a straightforward extension of standard notation: if $\Omega \subseteq \Omega'$ and A is an Ω':-algebra, we write $A\!\downarrow\Omega$ to denote the reduct Ω:-algebra $A\!\downarrow\iota$, where $\iota : \Omega \to \Omega'$ is the inclusion signature morphism.

Definition 6.18. If $\vartheta : \langle \Omega, E \rangle \to \langle \Omega', E' \rangle$ is an ET theory morphism and $A \in \mathrm{Talg}(\Omega, E)$, then $A\!\uparrow\vartheta$ is the Ω':-algebra defined by $A\!\uparrow\vartheta = (T_{\Omega'(A)}/\approx(\vartheta, A))\!\downarrow\Omega'$.

Proposition 6.19. *Let ϑ and A be as in Definition* 6.18; *$A\!\uparrow\vartheta$ is a ϑ-free extension of A.*

Proof. It is sufficient to show the existence of an Ω:-morphism $\eta_A : A \to A\!\uparrow\vartheta\!\downarrow\vartheta$ that satisfies the condition in Definition 6.16. Let $\eta_A =_{\text{def}} \iota_{\eta_A} \circ \pi_{\eta_A}$, where, recalling

Proposition 6.13, $\pi_{\eta_A}: A \to A/{\approx}(A, \vartheta)$ is the natural projection, and $\iota_{\eta_A}: A/{\approx}(A, \vartheta) \to A{\uparrow}\vartheta{\downarrow}\vartheta$ is defined by $\iota_{\eta_A}([a]_{\approx(A,\vartheta)}) = [a]_{\approx(\vartheta,A)}$. Clearly, $\mathrm{Ker}(\eta_A) = \mathrm{Ker}(\pi_{\eta_A}) = {\approx}(A, \vartheta)$, thus $A{\uparrow}\vartheta$ is a ϑ-extension of A. Let B and $\phi: A \to B{\downarrow}\vartheta$ be as in Definition 6.16; if an Ω':-morphism $\phi^{\#}: A{\uparrow}\vartheta \to B$ exists that satisfies the commutativity condition in Definition 6.16, for the given η_A, then the map $\phi^{\#}: |A{\uparrow}\vartheta| \to |B|$ must satisfy the condition: $\phi^{\#}([a]_{\approx(\vartheta,A)}) = \phi(a) \forall a \in |A|$. But this condition, together with $B \in \mathrm{Talg}(\Omega', E')$ and the standrd Ω'-morphism condition on $\phi^{\#}: |A{\uparrow}\vartheta| \to |B|$, *define* an Ω':-morphism $\phi^{\#}: A{\uparrow}\vartheta \to B$; this is then the unique Ω':-morphism that meets that commutativity condition, for the given η_A. □

Free extension is turned into a functor in the usual way; if $\vartheta: \langle \Omega, E\rangle \to \langle \Omega', E'\rangle$ is an ET theory morphism and $\phi: A \to B$ is an Ω:-morphism, then $\phi{\uparrow}\vartheta: A{\uparrow}\vartheta \to B{\uparrow}\vartheta$ is the unique Ω':-morphism such that $\phi{\uparrow}\vartheta{\downarrow}\vartheta \circ \eta_A = \eta_B \circ \phi$, where η_A, η_B are defined as in the above proof. Consistent with Propositions 6.17 and 6.19, $\_{\uparrow}\vartheta: \mathrm{TAlg}(\Omega, E) \to \mathrm{TAlg}(\Omega', E')$ is referred to as "the" ϑ-free functor from $\mathrm{TAlg}(\Omega, E)$ into $\mathrm{TAlg}(\Omega', E')$ that is determined by the ET theory morphism $\vartheta: \langle \Omega, E\rangle \to \langle \Omega', E'\rangle$. Clearly, this functor is left adjoint to $\_{\downarrow}\vartheta: \mathrm{TAlg}(\Omega', E') \to \mathrm{TAlg}(\Omega, E)$, with η_A the components of the unit of the adjunction.

The well-known link between freeness and initiality says that ϑ-free extension preserves initiality. Saying more about this link requires some notation, which "explains" some terminology in Definitions 6.3 and 6.4.

First, we generalize our last notational convention as follows: if $\Omega \subseteq \Omega'$ and A is an Ω':-algebra, then $A{\downarrow}^{\nu}\Omega$ denotes $A{\downarrow}^{\nu}\iota$, the Ω_{ν}-generated ι-reduct of A, where $\iota: \Omega \to \Omega'$ is the inclusion signature morphism. For the special case $\Omega = \Omega'$ (ι the identity), we adopt the following terminology: if A is an Ω:-algebra, then we say that A is Ω_{ν}-generated iff $A = A{\downarrow}^{\nu}\Omega$.

Proposition 6.20. *If $\vartheta: \langle \Omega, E\rangle \to \langle \Omega', E'\rangle$ is an ET theory morphism and $A \in \mathrm{Talg}(\Omega, E)$, then*

(i) *the $\Omega'(A)$:-algebra $A{\uparrow}\vartheta_A$ is initial in the class of ϑ_A-extensions of A (viewing A as an $\Omega(A)$:-algebra),*

(ii) *$A{\uparrow}\vartheta$ is initial in the class of ϑ-extensions of A iff $A{\uparrow}\vartheta$ is Ω'_0-generated,*

(iii) *$A{\uparrow}\vartheta$ is Ω'_0-generated if A is Ω_0-generated.*

Proof. (i) Immediate from Definitions 6.12 and 6.18.

(ii) (*Outline*) If B is a ϑ-extension of A, then Definitions 6.14, 6.16, and Proposition 6.19 entail the existence of an Ω':-morphism $\psi^{\#}: A{\uparrow}\vartheta \to B$. If $A{\uparrow}\vartheta$ is Ω'_0-generated, there may exist at most one map from $|A{\uparrow}\vartheta|$ to $|B|$ that satisfies the Ω':-morphism condition. Conversely, if $A{\uparrow}\vartheta$ is not Ω'_0-generated, then there exists a ϑ-extension of A such that distinct Ω':-morphisms from $A{\uparrow}\vartheta$ into it exist: such a ϑ-extension of A is easily constructed by suitably extending $A{\uparrow}\vartheta$.

(iii) By construction, $A{\uparrow}\vartheta$ is Ω'_0-generated if, for all $a \in |A|$, there exists an Ω'-term in the equivalence class $[a]_{\approx(\vartheta,A)}$. By hypothesis, A is Ω_0-generated, thus, for all $a \in |A|$, $a = t^A$ for some Ω-term t, which entails that ϑt is an Ω'-term in $[a]_{\approx(\vartheta,A)}$. □

The question naturally arises as to whether stronger reducts also have left adjoints. The answer is in the positive, and indeed these are "the same" free functor, but restricted to the appropriate subclasses: define $\mathrm{TAlg}(\Omega, E, \nu)$ by the condition

$$A \in \mathrm{TAlg}(\Omega, E, \nu) \text{ iff } A \in \mathrm{TAlg}(\Omega, E) \text{ and } A \text{ is } \Omega_\nu\text{-generated}$$

then $\_\uparrow\vartheta : \mathrm{TAlg}(\Omega, E, \nu) \to \mathrm{TAlg}(\Omega', E')$ is left adjoint to $\_\downarrow^\nu\vartheta : \mathrm{TAlg}(\Omega', E') \to \mathrm{TAlg}(\Omega, E, \nu)$. Moreover, the following generalization of Proposition 6.20 (iii) is easily expected: for all ν, $A\uparrow\vartheta$ is Ω'_ν-generated if A is Ω_ν-generated.

The investigation of what relationship the results presented in this section establish between ET logic and the theory of institutions [13] is a subject of our current research. Clearly, ET logic gives rise to a liberal institution: it is an open question whether liberality is preserved by extensions of ET logic with sentences of the kind "data constraints" [13]. A somewhat more radical question is whether such extensions give rise to institutions altogether.

7. Confluence of ET rewrite rules

Results are presented in this section that relate to the interpretation of equations in ET formulae as rewrite rules. The results by Bergstra and Klop on confluence of conditional rewrite rules [2] generalize to ET rules: shortly, those confluence results can be lifted to ET rules by just considering the equations only, with no worry about type assignment formulae whatsoever.

7.1. ET rewriting systems

We adopt the notation of [2] to a large extent, but sometimes detach from that notation, for convenience. Our first definitions extend the notion of a conditional term rewriting system to make room for "type assignment rules" in addition to the usual rewrite rules. We distinguish rules from ground rules according to whether or not variables occur.

Definition 7.1. (i) An *ET rule* is either a *rewrite rule*, having form "$t \Rightarrow u$", or a *type assignment rule*, having form "$t : u$", with "t" and "u" terms of $T_\Omega(V)$;

(ii) A *ground ET rule* is an ET rule without variables;

(iii) A ground ET rule δ is an *instance* of ET rule η if $\delta = \rho\eta$ for some closed substitution $\rho : V \to |T_\Omega|$.

Let S denote an ET presentation interpreted as what we call an "ET rewriting system", which—extending the terminology of [2]—is actually a "type assignment and term rewriting system involving conditional type assignment and rewrite rules". This interpretation is as follows. Equational conclusions are interpreted as rewrite rules: the set of these rules, written $\mathbf{R}_{\mathbf{ur}}$, is also called "the unconditional rewrite part" of S. Like in [2], *it is hereafter assumed that the rewrite rules in* $\mathbf{R}_{\mathbf{ur}}$ *are left-linear and (weakly) non-ambiguous.* Extending the classification proposed in [2] to the ET case, we consider three possible ways of interpreting the equational conditions

as *rewrite conditions*: either convertibility, written "$\sim$", or existence of a common reduct, written "$\Downarrow$", or reflexive-transitive reduction, written "$\Rightarrow^*$". In correspondence with each of these three interpretations, the resulting ET rewriting system will be called of type I, II or III, respectively. An ET rewriting system of type II^1_n is a system of type III such that the right-hand side of every rewrite condition is a (ground) normal form of $\mathbf{R}_{ur}$.

The notion of an ET rewriting system is a proper extension of the usual notion of a conditional term rewriting system, as considered also in [2]; this is made precise by the following definition, which also generalizes the classification proposed in [2].

Definition 7.2. An *ET rewriting system* is a triple $\langle \Omega, V, \mathbf{R}\rangle$, where Ω is a one-sorted signature, V a set of variables, and $\mathbf{R}$ a set of *conditional ET rules* of the form "$t_1 @ u_1, \ldots, t_n @ u_n \therefore t \# u$", where t, u, etc. are terms on Ω and V, "$\#$" in the conclusion can be ":" or "$\Rightarrow$", and "@" can be ":" (type assignment conditions) or one of "$\sim$" or "$\Downarrow$" or "$\Rightarrow^*$" (rewrite conditions): only one of these three types of rewrite conditions is made use of in any given ET rewriting system, whereby the ET rewriting system is of type I or II or III, respectively, provided that $\mathbf{R}_{ur}$, the set of rewrite rules "$t \Rightarrow u$" that are conclusions of $\mathbf{R}$, satisfies left-linearity and weak non-ambiguity. If the ET rewriting system S is of type III and the right-hand side of every rewrite condition is a ground normal form of $\mathbf{R}_{ur}$, then S is said to be of type III_n.

We recall or establish a few other concepts and notational matters.

Definition 7.3. If B is a binary relation on $|T_\Omega|$, the *contextual closure* of B, written $B^=$, is the smallest binary relation on $|T_\Omega|$ that includes B and satisfies the following condition:

$$\langle t, u\rangle \in B^= \Rightarrow \langle C[t], C[u]\rangle \in B^= \quad \text{for every } \Omega\text{-context } C[\ ].$$

Definition 7.4. Let $\mathbf{Y}$ be a set of ground ET rules. Then

(i) $\Rightarrow_{\mathbf{Y}}$ (*one-step reduction relation*) is the contextual closure of the rewrite rules in $\mathbf{Y}$;

(ii) $\Rightarrow^*_{\mathbf{Y}}$ (*reduction relation*) is the reflexive-transitive closure of $\Rightarrow_{\mathbf{Y}}$;

(iii) $\sim_{\mathbf{Y}}$ (*convertibility*) is the symmetric closure of $\Rightarrow^*_{\mathbf{Y}}$;

(iv) $\mathbf{Y}^0$ is the union of $\Rightarrow^*_{\mathbf{Y}}$ and the set of type assignment rules

$$\{t{:}u \mid t \sim_{\mathbf{Y}} t',\ u \sim_{\mathbf{Y}} u',\ \text{for some } t'{:}u' \in \mathbf{Y}\}.$$

Notation 7.5. Let $\mathbf{Y}$ be as in Definition 7.4, and "@" be either "$\sim$" or "$\Downarrow$" or "$\Rightarrow$" or ":"; the notation $\mathbf{Y} \models t @ u$ then means:

(i) $t \sim_{\mathbf{Y}} u$, if "@" is "$\sim$";

(ii) t and u have a common reduct with respect to the rewrite rules of $\mathbf{Y}$, if "@" is "$\Downarrow$";

(iii) $t \Rightarrow^*_{\mathbf{Y}} u$, if "@" is "$\Rightarrow^*$";

(iv) $t{:}u \in \mathbf{Y}^0$, if "@" is ":".

We recall briefly the confluence results established by [2] for conditional term rewriting systems: if S is such a system, then S is confluent if it is of type I or III_n; systems of type II or III are not confluent in the general case. The latter, negative result also holds in our framework, since conditional term rewriting systems are a special case of ET rewriting systems. However, also the positive result, relating to systems of type I or III_n, does more generally hold for ET rewriting systems, as we are going to show. We start with the following definition (cf. Definition 2.4.1 in [2]).

Definition 7.6 (*Application of sets of conditional ET rules*). Let $\mathbf{X}$ be a set of ground conditional ET rules (i.e. conditional ET rules without variables) and $\mathbf{Y}$ be a set of ground ET rules. Then $\mathbf{X}(\mathbf{Y})$ (read "$\mathbf{X}$ applied to $\mathbf{Y}$") is the following set of ET rules:

$$\delta \in \mathbf{X}(\mathbf{Y}) \text{ iff } \delta \in \mathbf{Y} \text{ or there exists a conditional ET rule } t_1 @ u_1, \ldots, t_n @ u_n \therefore \delta \text{ in } \mathbf{X} \text{ such that } \mathbf{Y} \models t_i @ u_i \text{ for all } i = 1, \ldots, n.$$

Example 7.7. The following ET rewriting system of type III_n is the interpretation of an ET presentation of stacks of zeros, with top and pop operators limited to stacks having only two elements.

$$0 : \mathrm{nat}, \quad 0 : \mathrm{el}, \quad \mathrm{empty} : \mathrm{stack},$$

$$x : \mathrm{nat} \therefore S(x) : \mathrm{nat},$$

$$x : \mathrm{stack}, y : \mathrm{el} \therefore \mathrm{push}(x, y) : \mathrm{stack}$$

$$\#(\mathrm{empty}) \Rightarrow 0$$

$$x : \mathrm{stack}, y : \mathrm{el} \therefore \#(\mathrm{push}(x, y)) \Rightarrow S(\#(x))$$

$$x : \mathrm{stack}, y : \mathrm{el}, \#(x) \Rightarrow^* S(0) \therefore \mathrm{pop}(\mathrm{push}(x, y)) \Rightarrow x$$

$$x : \mathrm{stack}, y : \mathrm{el}, \#(x) \Rightarrow^* S(0) \therefore \mathrm{top}(\mathrm{push}(x, y)) \Rightarrow y$$

Let $\mathbf{X}$ be the set of ground instances of the previous conditional ET rules. Then Definition 7.6, with $\mathbf{Y} = \emptyset$, yields:

$$\mathbf{X}(\emptyset) = \{0 : \mathrm{nat}, 0 : \mathrm{el}, \mathrm{empty} : \mathrm{stack}, \#(\mathrm{empty}) \Rightarrow 0\};$$

$$\begin{aligned}\mathbf{X}(\mathbf{X}(\emptyset)) = \mathbf{X}(\emptyset) \cup \{&S(0) : \mathrm{nat}, \ \mathrm{push}(\mathrm{empty}, 0) : \mathrm{stack}, \ S(\#(\mathrm{empty})) : \mathrm{nat}, \\ &\mathrm{push}(\mathrm{empty}, \#(\mathrm{empty})) : \mathrm{stack}, \\ &\#(\mathrm{push}(\mathrm{empty}, 0)) \Rightarrow S(\#(\mathrm{empty})), \\ &\#(\mathrm{push}(\mathrm{empty}, \#(\mathrm{empty}))) \Rightarrow S(\#(\mathrm{empty}))\}.\end{aligned}$$

Note that the type assignment rule "$S(\#(\mathrm{empty})) : \mathrm{nat}$" is in $\mathbf{X}(\mathbf{X}(\emptyset))$ because the ground conditional ET rule "$\#(\mathrm{empty}) : \mathrm{nat} \therefore S(\#(\mathrm{empty})) : \mathrm{nat}$" is in $\mathbf{X}$, and moreover $\#(\mathrm{empty}) \sim_{\mathbf{X}(\emptyset)} 0$, $\mathrm{nat} \sim_{\mathbf{X}(\emptyset)} \mathrm{nat}$, and "$0 : \mathrm{nat}$" $\in \mathbf{X}(\emptyset)$.

On the contrary, the type assignment rule "$\#(\mathrm{empty}) : \mathrm{nat}$" is *not* in $\mathbf{X}(\mathbf{X}(\emptyset))$, because no ground conditional ET rule exists in $\mathbf{X}$ that has conclusion "$\#(\mathrm{empty}) : \mathrm{nat}$".

Notation 7.8. $\mathbf{X}^{k+1}(\mathbf{Y}) = \mathbf{X}^k(\mathbf{X}(\mathbf{Y}))$ inductively, with $\mathbf{X}^0(\mathbf{Y}) = \mathbf{Y}$. Moreover, if $\langle \Omega, V, \mathbf{R}\rangle$ is an ET rewriting system, we let $\mathfrak{R}$ denote the set of conditional ET rules that are ground instances of $\mathbf{R}$.

Definition 7.9. If $S = \langle \Omega, V, \mathbf{R}\rangle$ is an ET rewriting system, the *set of ground ET rules of* S is $\mathrm{GET}_S =_{\mathrm{def}} \bigcup_{n\in\mathbf{N}} \mathfrak{R}^n(\emptyset)$.

Definition 7.10. If $S = \langle \Omega, V, \mathbf{R}\rangle$ is an ET rewriting system of type III, then the *intermediate reductions* $\Rightarrow_k$ on the ground terms of S, with $k \in \mathbf{N}$, are inductively defined as follows, where each $\Rightarrow_k^*$ is the reflexive-transitive closure of $\Rightarrow_k$, $\Rightarrow_0 = \emptyset$, and $C[t] \Rightarrow_{k+1} C[u]$ ($C[\ ]$ an Ω-context) iff there exists a conditional ET rule δ in $\mathfrak{R}$ such that

(i) $t \Rightarrow u$ is the conclusion of δ, and

(ii) $t' \Rightarrow_k^* u'$ for every rewrite condition $t' \Rightarrow^* u'$ in δ, and

(iii) for every type assignment condition $t' : u'$ in δ, there exists a type assignment rule $t'' : u''$ in GET_S such that $t' \sim_{\mathrm{GET}_S} t''$ and $u' \sim_{\mathrm{GET}_S} u''$.

Definition 7.10 applies to ET rewriting systems of type III only, but it is clear how to modify it for those of type I and II.

Theorem 7.11. *$t \Rightarrow_{\mathrm{GET}_S} u$ iff $t \Rightarrow_k u$ for some k.*

Proof. We show by induction on n that $\Rightarrow_{\mathfrak{R}^n(\emptyset)} \subseteq \Rightarrow_n$, which entails $\Rightarrow_{\mathrm{GET}_S} \subseteq \bigcup_{n\in\mathbf{N}} \Rightarrow_n$ by Definition 7.9. The basis is immediate, as $\mathfrak{R}^0(\emptyset) = \emptyset$. The inductive step follows. $C[i] \Rightarrow_{\mathfrak{R}^{n+1}(\emptyset)} C[u]$ iff $C[t] \Rightarrow_{\mathfrak{R}^n(\emptyset)} C[u]$ (with $\Rightarrow_{\mathfrak{R}^n(\emptyset)} \subseteq \Rightarrow_n$ by induction hypothesis, and $\Rightarrow_n \subseteq \Rightarrow_{n+1}$ by Definition 7.10) or there exist a conditional ET rule "$t_1 @ u_1, \ldots, t_m @ u_m \therefore t \Rightarrow u$" in $\mathfrak{R}$ such that, for $i = 1, \ldots, m$:

Case 1: "$t_i @ u_i$" = "$t_i : u_i$" and there exists a "$t' : u'$" $\in \mathfrak{R}^n(\emptyset)$ with $t_i \sim_{\mathfrak{R}^n(\emptyset)} t'$ and $u_i \sim_{\mathfrak{R}^n(\emptyset)} u'$; on the other hand $\mathfrak{R}^n(\emptyset) \subseteq \mathrm{GET}_S$ by Definition 7.9, hence $t_i \sim_{\mathrm{GET}_S} t'$ and $u_i \sim_{\mathrm{GET}_S} u'$, thus falling in case (iii) of Definition 7.10.

Case 2: "$t_i @ u_i$" = "$t_i \Rightarrow^* u_i$" and $t_i \Rightarrow_{\mathfrak{R}^n(\emptyset)^*} u_i$; by induction hypothesis $\Rightarrow_{\mathfrak{R}^n(\emptyset)} \subseteq \Rightarrow_n$, hence $t_i \Rightarrow_n^* u_i$, thus falling in case (ii) of Definition 7.10.

Conversely, again by induction on n we show that $\Rightarrow_n \subseteq \Rightarrow_{\mathrm{GET}_S}$. Clearly $\Rightarrow_0 = \emptyset \subseteq \Rightarrow_{\mathrm{GET}_S}$. Assuming $\Rightarrow_n \subseteq \Rightarrow_{\mathrm{GET}_S}$, we show that $\Rightarrow_{n+1} \subseteq \Rightarrow_{\mathrm{GET}_S}$.

Now, $C[t] \Rightarrow_{n+1} C[u]$ iff $\exists \delta =$ "$t_1 @ u_1, \ldots, t_m @ u_m \therefore t \Rightarrow u$" $\in \mathfrak{R}$ such that

(1) $t_i \Rightarrow_n^* u_i$ for every rewrite condition "$t_i \Rightarrow^* u_i$" in δ, and

(2) $t_i \sim_{\mathrm{GET}_S} t'$ and $u_i \sim_{\mathrm{GET}_S} u'$ for some type assignment rule "$t' : u'$" $\in \mathrm{GET}_S$, for every type assignment condition in δ.

If we show that there exists a natural number r such that

(i) $t_i \Rightarrow_{\mathfrak{R}^r(\emptyset)} u_i$, for every rewrite condition "$t_i \Rightarrow^* u_i$" in δ, and

(ii) $t_i \sim_{\mathfrak{R}^r(\emptyset)} t'$ and $u_i \sim_{\mathfrak{R}^r(\emptyset)} u'$ for some type assignment rule "$t' : u'$" $\in \mathfrak{R}^r(\emptyset)$, for every type assignment condition in δ,

then we can infer $C[t] \Rightarrow_{\mathfrak{R}^{r+1}(\emptyset)} C[u]$, which implies $C[t] \Rightarrow_{\mathrm{GET}_S} C[u]$. Let $r =_{\mathrm{def}} \max(s_1, \ldots, s_m)$, where, for $i = 1, \ldots, m$, s_i is defined as follows:

Case 1: "$t_i \Rightarrow^* u_i$" is a rewrite condition in δ; $t_i \Rightarrow_n^* u_i$ iff there exists a derivation $z_0 \Rightarrow_n z_1 \Rightarrow_n \cdots \Rightarrow_n z_h$, with $z_0 = t_i$ and $z_h = u_i$ syntactically, for some $h \geqslant 0$; let $s_i =_{\text{def}} 0$ if $h = 0$, otherwise by induction hypothesis $z_j \Rightarrow_{\text{GET}_S} z_{j+1}$ for $j = 0, \ldots, h-1$; then for each such j, $\exists n_j\, z_j \sim_{\mathfrak{R}^{n_j}(\emptyset)} z_{j+1}$: let $s_i =_{\text{def}} \max(n_0, \ldots, n_{h-1})$ in this case.

Case 2: "$t_i : u_i$" is a type assignment condition in δ; by (2) above, $t_i \sim_{\text{GET}_S} t'$ and $u_i \sim_{\text{GET}_S} u'$ for some type assignment rule "$t' : u'$" $\in \text{GET}_S$. Moreover, "$t' : u'$" $\in \text{GET}_S$ iff "$t' : u'$" $\in \mathfrak{R}^h(\emptyset)$ for some h. Now, $t_i \sim_{\text{GET}_S} t'$ iff by a finite number of rewritings in GET_S $t_i \,{}^*\!\Leftrightarrow^*\, z_1 \,{}^*\!\Leftrightarrow^*\, z_2 \,{}^*\!\Leftrightarrow^* \cdots {}^*\!\Leftrightarrow^*\, t'$, for some finite sequence $z_1, z_2, \ldots$, where each occurrence of ${}^*\!\Leftrightarrow^*$ is either $\Rightarrow^*_{\text{GET}_S}$ or its converse ${}_{\text{GET}_S}^*\!\Leftarrow$ indifferently. Clearly, from $\text{GET}_S = \bigcup_{n \in \mathbf{N}} \mathfrak{R}^n(\emptyset)$ it follows that for every derivation $w_1 \Rightarrow_{\text{GET}_S} \cdots \Rightarrow_{\text{GET}_S} w_k$, a number m exists such that these rewritings are also rewritings in $\mathfrak{R}^m(\emptyset)$. Thus, a number h_{t_i} exists such that the rewritings $t_i \,{}^*\!\Leftrightarrow^*\, z_1 \,{}^*\!\Leftrightarrow^*\, z_2 \,{}^*\!\Leftrightarrow^* \cdots {}^*\!\Leftrightarrow^*\, t'$ are also rewritings in $\mathfrak{R}^{h_{t_i}}(\emptyset)$. By the same reasoning, $u_i \sim_{\text{GET}_S} u'$ iff a similarly defined number h_{u_i} exists. *Let* $s_i =_{\text{def}} \max(h, h_{t_i}, h_{u_i})$. The check that the so defined number r satisfies (i) and (ii) above completes the proof. □

7.2. Confluence of ET rewriting systems of type III$_n$

Known results on term rewriting systems and our assumptions ensure that $\mathbf{R}_{\text{ur}}$, the unconditional rewrite part of S, is confluent. Since, in general, the rewrite rules of GET_S are strictly contained in (the ground instances of) $\mathbf{R}_{\text{ur}}$, we cannot say the same about GET_S. But if we can show that in the elementary diagram

$$\begin{array}{ccccc} A \Rightarrow_{\mathbf{R}_{\text{ur}}} B & & A \Rightarrow_{\text{GET}_S} B & & B \\ \Downarrow_{\mathbf{R}_{\text{ur}}} \quad {}_*\!\Downarrow_{\mathbf{R}_{\text{ur}}} & \text{whenever} & \Downarrow_{\text{GET}_S} & \text{then also} & {}_*\!\Downarrow_{\text{GET}_S} \\ C \Rightarrow^*_{\mathbf{R}_{\text{ur}}} D & & C & & C \Rightarrow^*_{\text{GET}_S} D \end{array}$$

then also the confluence of GET_S follows as a direct consequence of that of $\mathbf{R}_{\text{ur}}$, since in the GET_S diagram we will use the same reductions as in the elementary $\mathbf{R}_{\text{ur}}$ diagram (see Lemma 7.12). Taking Theorem 7.11 into account, the same statement is obtained if we show the following.

Lemma 7.12. *If $C \,{}_m\!\Leftarrow A \Rightarrow_n B$, then the common reduct D obtained from the elementary diagram of $\mathbf{R}_{\text{ur}}$ that is determined by these two reduction steps satisfies the following condition : $B \Rightarrow^*_m D$ and $C \Rightarrow^*_n D$.*

Proof. The proof of this lemma goes as in [2], by induction to $m+n$, and is indeed identical except for showing, in the inductive step, that the reduction $C \Rightarrow^* D$ is an intermediate reduction $C \Rightarrow^*_n D$. The case $n+m=0$ is trivial, since $\Rightarrow_0$ is the empty relation. Assume the lemma is true for all n, m such that $n+m \leqslant k$. Let $n+m=k+1$. Say $n>0$. The interesting case is when A is a redex, $A = \rho(t)$, that

contains a proper subredex S which is reduced in the step $A \Rightarrow_m C$, as shown in the following diagram.

$$\begin{array}{ccc} A = \rho t = \_H[S]\_ & \Rightarrow_n & \_S\_S\_S\_ = \rho u = B \\ \Downarrow_m & & {}_*\Downarrow_m \\ C = \_H[S']\_ & \overset{?}{\Rightarrow}{}^*_n & \_S'\_S'\_S'\_ = D. \end{array}$$

In the reduction $B \Rightarrow^* D$, where copies of S are reduced, there is no problem: $B \Rightarrow^*_m D$. We now show that $C \Rightarrow^*_n D$. Let the reduction $A \Rightarrow_n B$ be generated by conditional ET rule δ under the closed substitution ρ. Let $\rho t \Rightarrow \rho u$ be the conclusion of $\rho\delta$ (so $A = \rho t$ syntactically). From Definition 7.10 it follows that $\rho t' \Rightarrow^*_{n-1} u'$ (where u' is a ground normal form of $\mathbf{R}_{ur}$) for every rewrite condition $\rho t' \Rightarrow^* u'$ in $\rho\delta$, and that for every type assignment condition $\rho t' : \rho u'$ in $\rho\delta$ there exists a type assignment rule $t'' : u''$ in GET_S such that $\rho t' \sim_{\mathrm{GET}_S} t''$ and $\rho u' \sim_{\mathrm{GET}_S} u''$. The weak non-ambiguity of the unconditional rewrite rules of $\mathbf{R}_{ur}$ entails that in the reduction step $A \Rightarrow_m C$ the redex S is contained in ρx for some variable x in t: say $\rho x = H[S]$ for some context $H[\ ]$. Let ρ' be the closed substitution such that $\rho' x = H[S']$, with S' the contractum of S in the reduction step $A \Rightarrow_m C$, whereas $\rho' y = \rho y$ for every other variable y. We have to show that (i) $\rho' t' \Rightarrow^*_{n-1} u'$ for every rewrite condition $\rho' t' \Rightarrow^* u'$ in $\rho'\delta$, and that (ii) for every type assignment condition $\rho' t' : \rho' u'$ in $\rho'\delta$ there exists a type assignment rule $t'' : u''$ in GET_S such that $\rho' t' \sim_{\mathrm{GET}_S} t''$ and $\rho' u' \sim_{\mathrm{GET}_S} u''$. (i) is shown as in Theorem 3.5 of [2], (ii) holds because $\rho t' \sim_{\mathrm{GET}_S} t''$ and $\rho u' \sim_{\mathrm{GET}_S} u''$ for some $t'' : u''$ in GET_S, as argued above, and moreover $\rho t' \Rightarrow_{\mathrm{GET}_S} \rho' t'$, $\rho u' \Rightarrow_{\mathrm{GET}_S} \rho' u'$, since $S \Rightarrow_m S'$ and by Theorem 7.11 this gives $S \Rightarrow_{\mathrm{GET}_S} S'$: then $\rho' t' \sim_{\mathrm{GET}_S} t''$ and $\rho' u' \sim_{\mathrm{GET}_S} u''$. □

7.3. Confluence of ET rewriting systems of type I

To extend the result from conditional term rewriting systems (see Theorem 3.2 and Corollary 3.3 in [2]) to ET rewriting systems, we first define when does a closed substitution satisfy the assumption of a rule in an ET rewriting system of type I.

Definition 7.13. Let $S = \langle \Omega, V, \mathbf{R} \rangle$ be an ET rewriting system of type I, and $\rho : V \to |T_\Omega|$ a closed substitution. We say that ρ satisfies the assumption of a conditional ET rule $\delta \in \mathbf{R}$ if:

(i) $\rho t' \sim_{\mathrm{GET}_S} \rho u'$ for every rewrite condition $t' \sim u'$ in δ, and

(ii) for every type assignment condition $t' : u'$ in δ, there exists a type assignment rule $t'' : u''$ in GET_S such that $\rho t' \sim_{\mathrm{GET}_S} t''$ and $\rho u' \sim_{\mathrm{GET}_S} u''$.

Lemma 7.14. *Let $S = \langle \Omega, V, \mathbf{R} \rangle$ be an ET rewriting system of type I, δ a conditional ET rule in $\mathbf{R}$ that has conclusion $t \Rightarrow u$, and ρ a closed substitution. Then $\rho t \Rightarrow \rho u \in \mathrm{GET}_S$ iff ρ satisfies the assumption of δ.*

Proof. From $\mathrm{GET}_S = \bigcup_{n \in \mathbf{N}} \mathfrak{R}^n(\emptyset)$ it follows that there exists a natural number k such that $\rho t'$ and $\rho u'$ are convertible with reductions in $\mathfrak{R}^k(\emptyset)$, for every rewrite condition $t' \sim u'$ in δ. □

Theorem 7.15. If S is an ET rewriting system of type Γ, then S is confluent.

Proof. With reference to the construction of the elementary diagram generated by two diverging reduction steps, $C_{\mathrm{GET}_S} \Leftarrow A \Rightarrow_{\mathrm{GET}_S} B$, let the reduction $A \Rightarrow_{\mathrm{GET}_S} B$ be generated by the conditional ET rule δ under the closed substitution ρ (then ρ satisfies the assumption of δ). By using the same symbols and notation as in the proof of Lemma 7.12, we have to show that the closed substitution ρ' such that $\rho' x = H[S']$, with S' the contractum of S in the reduction step $A \Rightarrow_{\mathrm{GET}_S} C$, and $\rho' y = \rho y$ for every other variable y, satisfies the assumption of δ (see diagram).

$$\begin{array}{ccc} A = \rho t = \_H[S]\_ & \Rightarrow_{\mathrm{GET}_S} & \_S\_S\_S\_ = \rho u = B \\ \Downarrow_{\mathrm{GET}_S} & & *\Downarrow_{\mathrm{GET}_S} \\ C = \_H[S']\_ & \overset{?}{\Rightarrow}^{*}_{\mathrm{GET}_S} & \_S'\_S'\_S'\_ = D \end{array}$$

The proof that $\rho' t' \sim_{\mathrm{GET}_S} \rho' u'$ for every rewrite condition $t' \sim u'$ in δ is as follows: $\rho' t' \,{}_{\mathrm{GET}_S}\!\Leftarrow \rho t' \sim_{\mathrm{GET}_S} \rho u' \Rightarrow_{\mathrm{GET}_S} \rho' u'$, where $\rho t' \Rightarrow_{\mathrm{GET}_S} \rho' t'$ by reducing the redex S in $\rho x = H[S]$, and similarly for $\rho u' \Rightarrow_{\mathrm{GET}_S} \rho' u'$. For type assignment conditions, the reasoning is analogous to that showing fact (ii) in the proof of Lemma 7.12. □

8. Related work

Similar ideas have been independently investigated by Mosses [22], Poigné [23] and Smolka [27].

Mosses *unified algebras* share many motivations, concepts and pragmatics with our type algebras, in particular they also are one-sorted total algebras with classifications (corresponding to our typing relation) and cater for dependent types in a straightforward way. Unified algebras are richer structures indeed, precisely distributive lattices with a bottom, which enables the representation of partial (possibly non-strict) operations in a total-algebra setting. Classification in unified algebras is viewed as a special case of the inclusion partial order. This entails reflexivity of classification of elements, a requirement that our approach does not enforce in the general case. Higher-order classification is an expressive possibility that unified algebras seem not to have: it is a question whether these structures can be generalized to allow such a facility. Mosses' work has been a source of inspiration to ours in at least two respects: the recognition of the greater simplicity of a total-algebra semantics, and the construction of the adjunction presented in Section 6—a prerequisite for the semantics of modular specification.

Our work shares much with that of Poigné [23] as well, in particular the original motivations (a wish of semantical uniformity for exception handling), the representation of partiality in a total-algebra setting, and the relevance of dependent types. To us, the main differences with respect to our approach seem to be: (1) usage of Scott's theory of partiality, (2) separation between typechecking and equational

specification, and (3) restriction on types, which we would like to find in the pragmatics, if at all, rather than built-in from the foundations.

Less significant than what it might seem is instead the unavailability in ET logic of subsorting, say "$t < u$" with t and u types, as a primitive (atomic) formula. Of course subsorting, or subtyping, is a very useful specification facility. The advantage of *not* having subtyping in the logic is that several, different forms of it can be introduced by appropriate ET theories, each form proving its merits in the appropriate context; so, for instance, one may like to have a subtype pre-order in some cases, or a partial order in others. Moreover, translations of (conditional) equational order-sorted logic and logics of partial algebras in ET logic are available [20], that yield complete calculi for such logics.

9. Conclusions

Equational type logic enjoys several nice properties: availability of a sound and complete calculus, standard construction of the initial model, straightforward extension of universal algebra notions and results. We have investigated formal notions of reduction and extension, motivated by the desirability of specification structuring facilities, which are most relevant to application. Further work remains to be done, e.g. in connection with the theory of institutions. The expressiveness of equational type logic in other frameworks, such as category theory, order-sorted logics, partial algebras, is a subject of our current research [20]. On the computational side we have extended the confluence results of [2] to ET rewriting systems: what seems promising is the synergy between rewriting and type inference, which in these systems are put together in a simple form. There is much to study here, e.g. completion procedures, type assignment functions, termination.

We conclude with an invitation to further investigation: some of the potential developments have been just mentioned in the present paper.

Acknowledgment

We are grateful to the anonymous referees for their useful suggestions. To one in particular, we are indebted for a very acute review which greatly contributed to improving the technical and editorial qualities of this paper.

References

[1] H. Andreka, P. Burmeister and I. Nemeti, Quasivarieties of partial algebras—a unifying approach towards a two-valued model theory for partial algebras, Preprint Nr. 557, FB Mathematik und Informatik, TH Darmstadt, 1980.

[2] J.A. Bergstra and J.W. Klop, Conditional rewrite rules: confluence and termination, *J. Comput. Systems Sci.* **32**(3) (1986) 323-362.

[3] G. Bernot, M. Bidoit and C. Choppy, Abstract data types with exception handling: an initial approach based on a distinction between exceptions and errors, *Theoret. Comput. Sci.* **46**(1) (1986) 13-45.

[4] G. Birkhoff and J.D. Lipson, Heterogeneous algebras, *J. Combin. Theory* **8** (1970) 115-133.

[5] V. Breazu-Tannen and J. Gallier, Polymorphic rewriting conserves algebraic strong normalization and confluence, in: G. Ausiello, M. Dezani-Ciancaglini and S. Ronchi Della Rocca, eds., *Automata, Languages and Programming, Proc. 16th ICALP*, Lecture Notes in Computer Science **372** (Springer, Berlin, 1989) 137-150.

[6] M. Broy and M. Wirsing, Partial abstract types, *Acta Inform.* **18** (1982) 47-64.

[7] P. Burmeister, *A Model Theoretic Oriented Approach to Partial Algebras* (Akademie-Verlag, Berlin, 1986).

[8] R.M. Burstall and J.A. Goguen, Putting theories together to make specifications, in: *Proc. 5th Internat. Joint Conf. on Artificial Intelligence* (Kaufman, Cambridge, MA, 1977) 1045-1058.

[9] H. Ehrig and B. Mahr, *Fundamentals of Algebraic Specification* 1 (Springer, Berlin, 1985).

[10] K. Futatsugi, J.A. Goguen, J.-P. Jouannaud and J. Meseguer, Principles of OBJ2, in: *Proc. Symp. Principles of Programming Languages* (ACM, 1985) 52-66.

[11] J.A. Goguen, Abstract errors for abstract data types, in: *Proc. IFIP Working Conf. on Formal Description of Programming Concepts* (MIT, Cambridge, MA, 1977).

[12] J.A. Goguen, Order sorted algebras, semantics and theory of computation, Report 14, University of California at Los Angeles, 1978.

[13] J. Goguen and R. Burstall, Introducing institutions, in: E. Clarke and D. Kozen, eds., *Proc. Logics of Programming Workshop*, Lecture Notes in Computer Science **164** (Springer, Berlin, 1984) 221-256.

[14] J.A. Goguen and J. Meseguer, Models and equality for logical programming, in: H. Ehrig, R. Kowalski, G. Levi and U. Montanari, eds., *TAPSOFT '87, Volume* 2, Lecture Notes in Computer Science **250** (Springer, Berlin, 1987) 1-22.

[15] J.A. Goguen, J.W. Thatcher and E.G. Wagner, An initial algebra approach to the specification, correctness, and implementation of abstract data types, in: R. Yeh, ed., *Current Trends in Programming Methodology IV* (Prentice Hall, Englewood Cliffs, NJ, 1978) 80-149.

[16] W.S. Hatcher, *The Logical Foundations of Mathematics* (Pergamon Press, Oxford, 1982).

[17] P.J. Higgins, Algebras with a scheme of operators, *Math. Nachr.* **27** (1963) 115-132.

[18] V. Manca and A. Salibra, On the power of equational logic: applications and extensions, University of Pisa, Dip. Informatica, TR-19/88; to appear in: *Proc. Internat. Conf. on Algebraic Logic*, Budapest, August 8-14, 1988.

[19] V. Manca, A. Salibra and G. Scollo, On the nature of TELLUS, a typed equational logic look over uniform specification, in: A. Kreczmar and G. Mirkowska, eds., *Mathematical Foundations of Computer Science 1989*, Lecture Notes in Computer Science **379** (Springer, Berlin, 1989) 338-349.

[20] V. Manca, A. Salibra and G. Scollo, On the expressiveness of equational type logic, forthcoming.

[21] P. Martin-Löf, An intuitionistic theory of types: predicative part, in: H.E. Rose and J.C. Shepherdson, eds., *Logic Colloquium 1973* (North-Holland, Amsterdam, 1975) 73-118.

[22] P.D. Mosses, Unified Algebras and Institutions, in: *Proc. 4th IEEE Symp. on Logic In Computer Science*, Pacific Grove, CA (IEEE, 1989).

[23] A. Poigné, Partial algebras, subsorting and dependent types: prerequisites of error handling in algebraic specification, in : D. Sannella and A. Tarlecki, eds., *Recent Trends in Data Type Specification*, Lecture Notes in Computer Science **332** (Springer, Berlin, 1988) 208-234.

[24] H. Reichel, *Initial Computability, Algebraic Specifications, and Partial Algebras* (Clarendon Press, Oxford, 1987).

[25] T. Rus, HAS-hierarchy: a natural tool for language specification, *Fund. Inform.* 3(3) (1979) 269-294.

[26] G. Scollo, On the use of equational type logic for software engineering and protocol design, in: *1st Maghrebin Conf. on Artificial Intelligence and Software Engineering*, Constantine, Algeria (1989).

[27] G. Smolka, Type logic, in: *Proc. 6th ADT Workshop*, Berlin (1988) Abstract.